

Università degli studi di Roma "Tor Vergata"
Corso di Laurea in Ingegneria Informatica

**Alberi di decisione
per l'estrazione automatica
di terminologia da testi**

Relatore:
Dott. Roberto Basili

Correlatore:
Dott. Fabio Massimo Zanzotto

Laureando:
Bonaventura Coppola

Anno Accademico 2000-2001

Indice

1	Introduzione	4
2	Estrazione di terminologia di dominio	7
2.1	Il concetto di <i>termine</i>	7
2.2	Caratteristiche semantiche e strutturali dei termini	11
2.3	Modello esteso del dominio	12
2.4	Informazione endogena ed esogena.	13
2.5	Metodi di analisi	15
2.5.1	Analisi morfologica	16
2.5.2	Analisi sintattica	18
2.5.3	Costrutti linguistici ricorrenti	20
2.5.4	Associazioni lessicali nei termini multi-word	22
2.6	Probematiche di individuazione dei termini	23
2.6.1	Problemi di utilizzo dei costrutti ricorrenti	23
2.6.2	Varianti nei termini complessi	25
2.7	Metriche di valutazione per la TE	26
3	Apprendimento automatico	29
3.1	Il processo di apprendimento	29
3.2	Il contesto ambientale	30

3.3	La rappresentazione dell'ambiente	32
3.4	Apprendimento nei problemi di classificazione	33
3.5	Alberi di decisione	35
3.6	Il sistema di apprendimento C4.5	40
4	Apprendimento automatico di terminologia	42
4.1	Obiettivo specifico di questo studio	42
4.2	Il problema dei <i>non-termini</i>	43
4.3	Rappresentazione dell'informazione contestuale	44
4.4	Il problema della modellizzazione	47
5	Modelli di informazione contestuale	48
5.1	Formato dei dati disponibili	49
5.2	Formato richiesto da C4.5	50
5.3	Modello locale-lessicale (L)	51
5.4	Modello globale-lessicale (G)	52
5.5	Modello globale-sintattico (S)	53
5.6	Modello in frequenza (F)	54
5.7	modelli <i>ibridi</i>	55
6	Piattaforma sperimentale realizzata	57
6.1	Caratteristiche del corpus	57
6.2	Metodologia sperimentale	59
6.2.1	<i>n</i> -fold cross validation	59
6.2.2	Variazione progressiva degli esempi negativi	60
6.2.3	Formalizzazione della procedura generale	61
6.3	Software realizzato	62
6.3.1	Generatore di modelli " <i>filtro.pl</i> "	63
6.3.2	Iteratore computazionale " <i>iter.pl</i> "	63

6.3.3	Script di gestione dei lavori “ <i>batch</i> ”	64
7	Risultati sperimentali	65
7.1	Valutazioni preliminari	65
7.2	Variazione progressiva esempi negativi	66
7.3	Confronto con i modelli ibridi	69
8	Conclusioni	72
A	Codice sorgente commentato	74
A.1	Codice di <i>filtro.pl</i>	74
A.2	Codice di <i>iter.pl</i>	98
A.3	Esempio di script <i>batch</i>	103
	Bibliografia	106

Capitolo 1

Introduzione

Il presente studio si colloca nel contesto generale dell'elaborazione automatica del linguaggio naturale (Allen [13]). Nell'ambito di tale disciplina, verranno presentati argomenti teorici e risultati sperimentali la cui pertinenza specifica è propria dell'*estrazione automatica di terminologia* (Jacquemin [10]). Oggetto particolare di tale attività è l'analisi di una collezione di documenti di dominio al fine di selezionare tutti i *termini rappresentativi*.

Il concetto di termine assume un ruolo centrale nel corso di tutto lo studio e particolare attenzione verrà data ai suoi due aspetti principali: quello *semantico* e quello *grammaticale*. Relativamente all'aspetto semantico, il ruolo specifico del termine è quello di rappresentare dal punto di vista linguistico un *concetto* univoco e ben definito, nell'ambito di un determinato campo del sapere. Relativamente all'aspetto grammaticale, il termine si presenta come elemento dell'enunciato rispondente a requisiti formali (attinenti tanto alla sua struttura interna quanto al contesto di utilizzo) che, sebbene molto articolati, consentono l'impiego di tecniche di analisi automatiche o semi automatiche (Langley [11]).

Scopo specifico di questo studio è dunque quello di valutare nelle appli-

cazioni concrete di estrazione terminologica l'efficacia di una speciale sottoclasse di tali tecniche: quelle relative all'*apprendimento automatico induttivo*. L'apprendimento automatico sarà perciò il secondo grande campo di indagine oggetto di trattazione: verranno inizialmente presentati i suoi concetti di base, per poi rivolgere l'attenzione a particolari ambiti di attività (i *problemi di classificazione*) e a specifiche tecniche di apprendimento utilizzabili con successo in tali ambiti (gli *alberi di decisione*, Quinlan [14]).

Acquisiti gli strumenti operativi fondamentali, si cercherà sul piano sperimentale una precisa conferma delle ipotesi formulate nella prima parte. Oggetto dello studio sarà l'indagine di specifiche proprietà caratterizzanti la nozione di termine. Si analizzerà in particolare l'*informazione contestuale* associata ai termini, concetto al quale verrà dedicato ampio spazio per la sua importanza teorica e applicativa.

In relazione a quanto siamo andati finora delineando, la problematica più complessa da affrontare sul piano applicativo sarà la definizione di un modello di informazione contestuale rispondente a due requisiti: significatività e trattabilità computazionale. Dal punto di vista della prima, si cercherà un modello effettivamente in grado di astrarre la definizione generale di *termine* in un fissato contesto applicativo (nel nostro caso quello *legale*), mentre dal punto di vista del secondo si cercherà di codificare il modello stesso secondo un formalismo adatto alle tecniche di apprendimento induttivo utilizzate.

Appare naturale come un tale problema di modellazione ammetta più di una possibile soluzione. Si è cercato per questo di approfondire l'indagine ad un livello soddisfacente con la produzione di modelli *selettivi* per tipologie di informazione. Tale approccio ha permesso di confrontare separatamente gli aspetti *lessicali, sintattici e statistici* della nozione di termine.

La presentazione dei risultati quantitativi ottenuti in tali confronti speri-

mentali costituisce una fondata verifica dell'ipotesi generale di partenza, cioè l'efficacia dell'informazione contestuale nell'estrazione automatica di terminologia.

Nel capitolo 2 verrà presentata una panoramica generale dell'estrazione automatica di terminologia. Il capitolo 3 è dedicato alle tecniche di apprendimento automatico che verranno impiegate negli esperimenti, mentre il capitolo 4 costituisce una sintesi del particolare approccio applicativo su cui verte l'intero studio. La definizione formale dei modelli utilizzati è contenuta nel capitolo 5, mentre la loro valutazione sperimentale è oggetto dei capitoli 6 e 7. Seguono poi le conclusioni nel capitolo finale.

Capitolo 2

Estrazione di terminologia di dominio

Obiettivo del presente capitolo è fornire una panoramica generale degli argomenti legati all'estrazione di terminologia (*terminology extraction, TE*) in un dato dominio. Verranno introdotti innanzitutto i concetti fondamentali e le finalità della materia, per poi guardare con maggior dettaglio alle sue problematiche principali e alle tecniche computazionali sviluppate per superarle. Con riferimento alla qualità dell'informazione reperibile in un testo, verrà dato particolare risalto alla differenza tra *informazione endogena*, legata alla struttura dei termini, ed *informazione esogena*, riguardante invece il loro contesto sintattico e semantico. Tale distinzione costituisce il più importante punto cardine su cui verrà basato il lavoro successivo. Nel corso della trattazione verranno via via presentati anche ulteriori aspetti miranti a collocare la TE nel suo naturale contesto applicativo (le *TKB*), con particolare attenzione agli aspetti modellistici.

2.1 Il concetto di *termine*

Prima di entrare con maggior dettaglio negli aspetti specifici dell'estrazione di terminologia di dominio, conviene innanzitutto individuarne intuitivamen-

te il compito specifico. Scopo della TE è analizzare con opportuni metodi computazionali una collezione omogenea di testi appartenenti ad uno stesso dominio (denominata *corpus*) al fine di individuarne un insieme di termini rappresentativi. Tale operazione è normalmente finalizzata a due distinte applicazioni: l'acquisizione di conoscenza, mirata tipicamente alla costruzione di una base di conoscenza, oppure l'indicizzazione automatica dei testi, finalizzata ad applicazioni di recupero di documenti in una base dati. Pur trattandosi di due processi tradizionalmente accomunati nell'ambito della TE, si faccia però attenzione al fatto che, nel suggerirne l'associazione, si è implicitamente e volutamente fatto uso di un diverso significato della nozione "termine rappresentativo" in un caso e nell'altro. Tale ambiguità dirige la nostra attenzione direttamente al cuore della TE, che risiede appunto nel concetto stesso di *termine*. Mentre nel primo contesto delineato (basi di conoscenza) i termini sono considerati il fine dell'indagine stessa, nel secondo caso ne costituiscono chiaramente soltanto un mezzo. L'acquisizione di conoscenza terminologica è infatti una attività che mira precisamente alla costruzione e allo sviluppo di specifiche strutture informative, note come *basi di conoscenza terminologiche (terminological knowledge bases TKB)*, in grado di rappresentare non soltanto semplici dizionari di termini ma la ben più complessa rete concettuale indotta da questi. Viceversa, l'indicizzazione automatica ha lo scopo di assegnare a ciascun documento esaminato una serie di descrittori che rappresentino il contenuto *di quel documento specifico*, rendendo così più agevole il suo successivo reperimento. Tale attività, come verrà ulteriormente chiarito, è di pertinenza specifica della *information retrieval (IR)*. Sottolineando tale differenza si è voluta ulteriormente rimarcare la centralità del concetto di termine individuando tanto sul piano teorico quanto su quello applicativo la potenziale confusione derivante da una sua

non corretta definizione.

Al fine di risolvere questa ambiguità, in [7] si propone l'uso della parola *termine* per il primo dei due casi qui sopra delineati, e *keyword* per il secondo. Inoltre, i due vengono definiti formalmente come:

keyword: una sequenza di parole o unità lessicali che caratterizza un *documento* o un insieme di *documenti* in un database.

termine: una unità lessicale che caratterizza un *dominio* o un campo specifico, rappresentando un concetto usato stabilmente nel dominio stesso.

Per meglio comprendere tale distinzione si consideri l'esempio, in materia di intelligenza artificiale, di un articolo illustrante il funzionamento di un sistema esperto per il riconoscimento di funghi. In questo caso la parola *funghi* sarà ovviamente una keyword valida per quel particolare documento, ma non potrà certo essere considerata un termine specifico dell'intelligenza artificiale.

Prima di concentrarci ora sui termini e sulla TE è utile rilevare un ulteriore tratto distintivo tra i due ambiti applicativi che siamo andati fin qui delineando. Si noti come nell'uno e nell'altro caso la conoscenza richiesta in merito al dominio in esame sia profondamente differente. In applicazioni di information retrieval la dipendenza delle keywords (molto spesso costituite da singole parole) dal contesto sintattico e semantico espresso nel documento è sostanzialmente trascurabile e ciò comporta che le tecniche di indicizzazione possono perciò basare in massima parte il loro funzionamento su metodi puramente statistici, relativi ad esempio alla frequenza o alla particolare collocazione delle parole nel testo. In conseguenza di ciò, la maggioranza dei moderni sistemi di IR è basata su un *modello a spazio vettoriale*, nel quale una lista di keyword (sia essa l'indice di un documento o il frutto di una

interrogazione) è trattata come un vettore in uno spazio n -dimensionale, dove n è la cardinalità dell'insieme di tutte le keyword note nella base di dati. Nell'ambito di questo modello, l'attività di reperimento di un documento è sostanzialmente leggibile in termini di confronto vettoriale tra il vettore che esprime l'interrogazione e quelli relativi a tutti i documenti sui quali è svolta la ricerca. Basterà quindi definire sullo spazio una funzione di distanza vettoriale affinché il sistema risponda con una lista *ordinata* di documenti, fornendo per primi quelli con vettore-keyword a distanza minore dal vettore-interrogazione.

Passando ora ad applicazioni relative alle TKB, risulterà chiaro che l'analisi del corpus non può più fermarsi al conteggio di occorrenze o di collocazioni lessicali ricorrenti. Proprio per la definizione di termine introdotta, non è più ulteriormente trascurabile il fatto che la sua vera funzione è rappresentare sul piano linguistico un concetto ben definito. Ora, essendo la ricerca volta esattamente all'identificazione dei concetti e delle relazioni che essi stabiliscono tra loro (cioè la rete semantica indotta dal corpus), tale ricerca dovrà dotarsi di una serie di strumenti ben più raffinati di quelli puramente statistici, ed orientarsi all'individuazione di strutture linguistiche ricorrenti (*pattern*) non soltanto sul piano lessicale ma anche e soprattutto su quello sintattico.

L'opportunità di sviluppare tali strumenti di analisi implica necessariamente il ricorso a contenuti teorici specificamente pertinenti l'elaborazione automatica del linguaggio naturale.

2.2 Caratteristiche semantiche e strutturali dei termini

Dal punto di vista del loro ambito semantico, i termini sono usualmente distinti in *tecnici* e *scientifici*. Mentre i primi denotano tipicamente strumenti, manufatti, osservazioni, esperimenti o misure, i secondi denotano in generale concetti teorici in specifici domini scientifici. Entrambi i tipi di termine sono prevalentemente rappresentati nei dizionari in forma paradigmatica attraverso nomi o sintagmi nominali. Va tuttavia considerato che gli stessi termini potrebbero talvolta essere espressi in specifici contesti (dunque nell'uso corrente) da sintagmi di diverso tipo (verbali, aggettivali, ...). Ciò nonostante, nella maggior parte dei sistemi di TE, si tende generalmente a trascurare tale comportamento e si restringe l'analisi al solo ambito dei sintagmi nominali.

Altra importante classificazione dei termini, questa volta di tipo strutturale, distingue tra termini composti di un singolo lessema, che denominiamo *semplici*, o (*single-word*) e termini composti di più lessemi, che denominiamo *complessi*, o (*multi-word*). In un termine complesso conviene individuare un unico lessema cui attribuire un ruolo dominante (detto *testa*), mentre gli altri vengono genericamente considerati *modificatori* della testa stessa. A seconda che si ricada prevalentemente in un caso o nell'altro (semplici/complessi), i problemi di progettazione di un sistema di TE sono abbastanza diversi: i termini semplici, essendo generalmente polisemici, richiedono spesso una appropriata disambiguazione semantica ed una articolata analisi del contesto. I termini complessi sono molto meno frequentemente polisemici rispetto ai semplici, ma avendo struttura sintagmatica sono soggetti a molte possibili *varianti* (concetto che sarà definito più avanti) e la loro identificazione richiede in generale l'applicazione di analizzatori morfologici e sintattici o metodi

statistici per l'individuazione di collocazioni lessicali ricorrenti. A tale riguardo, è il caso di accennare brevemente che le applicazioni sperimentali considerate nel presente studio sono in qualche modo trasversali rispetto alle distinzioni fatte. Pur occupandoci infatti del caso generale di termini complessi, sarà data infatti particolare rilevanza alla modellazione del *contesto sintattico* del termine, mentre il problema della sua struttura interna sarà trattato esclusivamente sul piano teorico.

2.3 Modello esteso del dominio

Abbiamo finora considerato il significato e le caratteristiche generali dei termini da un punto di vista squisitamente concettuale. Ci preoccuperemo ora di dettagliare ulteriormente il contesto generale in cui si colloca il processo di TE, con particolare attenzione al dominio di riferimento a problematiche legate alla complessità intrinseca del linguaggio naturale.

Un importante dato di fatto da tener presente in ogni applicazione di TE è che il compito specifico della definizione di una terminologia pertinente a un dato dominio non può (ancora) prescindere in generale da un robusto intervento umano, sia esso di un terminologo o, più auspicabilmente, di un esperto di dominio. Compito di tale figura è quello di ispezionare manualmente liste di *candidati* termini estratti (semi)automaticamente dal corpus e di compilare dizionari *controllati* di tali termini.

In altre parole, i terminologi e gli esperti di dominio esplorano un *modello esteso del dominio*, costituito dal corpus e da dizionari preesistenti al fine di produrre una nuova terminologia.

Il compito generale dell'elaborazione di una nuova terminologia dovrà dunque rispettare tale schema generale, sia che esso venga condotto ma-

nualmente che con tecniche automatiche. In entrambi i casi, tale processo presenta due aspetti generali. Il primo prevede l'applicazione al testo di vincoli sintattici o semantici che isolino un sottoinsieme di termini candidati, il secondo comporta invece il confronto di tali candidati con quelli presenti nel dizionario controllato. A causa della elevata complessità intrinseca del linguaggio naturale, entrambi gli aspetti risultano notevolmente sofisticati. Il primo deve confrontarsi in particolare con la complessità della struttura grammaticale mostrata comunemente dai termini, mentre il secondo con la notevole molteplicità di forme grammaticali e lessicali distinte che *un medesimo termine* può assumere. Tali forme, note nel complesso con il nome di *varianti*, sono originate da fenomeni linguistici di tipo generale e presenti in tutte le lingue, noti come *flessione* e *derivazione*. È opportuno sottolineare a tale proposito che il processo di riconoscimento delle varianti e della loro riconduzione all'unico termine di riferimento è della massima importanza in TE, soprattutto quando questa sia finalizzata all'alimentazione di una TKB. In tale specifico caso infatti l'aspetto puramente lessicale della terminologia passa nettamente in secondo piano rispetto a quello semantico, nel quale la funzione propria del termine è quella di individuare univocamente un concetto.

2.4 Informazione endogena ed esogena.

Gli argomenti trattati nel seguito del presente capitolo tenteranno di mettere in evidenza la grande varietà di aspetti, di metodi e di problemi legati alle applicazioni di estrazione terminologica. Ciò che ora preme maggiormente sottolineare è invece la motivazione di fondo di tale varietà. Si vuole cioè osservare che, sebbene difficilmente ottenibile con mezzi automatici, l'in-

formazione terminologica effettivamente presente in un corpus è in generale estremamente ricca, tanto sul piano della quantità che su quello della qualità.

Relativamente al complesso di tale informazione, una distinzione estremamente rilevante ai fini del presente studio è quella tra informazione *endogena* ed informazione *esogena*, quest'ultima nota anche come informazione *contestuale* o *external evidence*. L'elemento di confine tra queste due tipologie di informazione terminologica è costituito dal termine stesso. Relativamente ad esso, definiamo quindi informazione endogena quella relativa alla sua struttura *interna*, dove per struttura intendiamo il complesso degli aspetti sintattico-grammaticali e semantici che concorrono nella determinazione del termine in senso assoluto. Definiamo viceversa *esogena* l'informazione relativa alle relazioni che il termine mantiene con il suo contesto, intendendo anche qui comprendere l'insieme degli aspetti sintattici e semantici. Appare evidente come la linea di demarcazione che si sta cercando di tracciare acquisti realmente spessore solo nel passaggio da termini semplici a termini complessi. Soprattutto dal punto di vista endogeno, infatti, la struttura sintagmatica di un termine complesso si presta in modo notevolmente più marcato ad essere oggetto di fenomeni che, come vedremo, daranno luogo a numerosi problemi sul piano applicativo.

La tendenza generale delle ricerche nell'ambito della terminologia computazionale sembra aver nettamente privilegiato tecniche di ispezione dei corpora basate sulla ricerca di informazione endogena. Alcuni studi condotti sull'informazione contestuale hanno inoltre privilegiato l'aspetto semantico lasciando comunque in secondo piano quello sintattico, come sarà avvenuto nel caso dei pattern linguistici, che saranno descritti tra i metodi di analisi.

Oggetto particolare di questo studio, invece, è proprio l'indagine del *contesto sintattico* relativo ai candidati termini individuati nel corpus. Siamo

perciò interessati dal punto di vista applicativo ad estrarre dal testo esclusivamente l'informazione esogena riguardante la *dipendenza sintattica* del candidato termine da eventuali altre parole nelle immediate vicinanze.

L'argomento specifico dell'informazione contestuale sarà dunque ulteriormente approfondito nella parte sperimentale, in cui occorrerà definirne concretamente un modello trattabile dal punto di vista computazionale.

2.5 Metodi di analisi

Definito il contesto generale in cui si colloca il processo di TE, procediamo ora ad una breve panoramica delle tecniche tradizionalmente utilizzate per la concreta individuazione dei (candidati) termini. I metodi di analisi utilizzati possono suddividersi in *simbolici* e *subsimbolici*. Una analisi simbolica è tesa a stabilire la struttura grammaticale e sintattica del testo, cercando di individuare gli elementi costitutivi in base alle categorie proprie del linguaggio in cui è stato scritto. Presuppone dunque la conoscenza delle regole grammaticali specifiche di una data lingua, al fine di riconoscerne l'uso all'interno del testo. Tipici strumenti di questo tipo sono gli analizzatori morfologici e sintattici (*parser*). I metodi subsimbolici affrontano invece l'analisi con metodi di indagine non direttamente legati alla struttura grammaticale del testo, basandosi dunque su strumenti statistici, probabilistici o derivanti dalla teoria dell'informazione. Esempi di questo tipo sono le tecniche basate sulla ricerca di associazioni lessicali nei termini complessi e/o di pattern linguistici presenti nel contesto, aspetti che verranno entrambi presentati e sviluppati nel seguito.

Metodi simbolici e subsimbolici sono spesso utilizzati in modo congiunto con buoni risultati. È pratica comune, ad esempio, l'uso di un filtro statistico

a valle di un analizzatore sintattico, in modo da selezionare tra i candidati termini solo quelli soddisfacenti determinati vincoli di frequenza, siano essi riferiti a tutto il corpus o ad un ambito locale, avendo in quest'ultimo caso fissato in precedenza una finestra della dimensione voluta.

2.5.1 Analisi morfologica

Cenni di morfologia

La morfologia è la parte della linguistica dedicata alla descrizione e allo studio della struttura delle parole. Inoltre indaga i meccanismi secondo i quali le unità portatrici di significati semplici si organizzano in significati più complessi. Questa organizzazione infatti implica la formazione o trasformazione delle parole, cioè la determinazione della loro forma, in base a quella che deve essere la loro funzione nella sintassi del discorso. Il termine “morfologia” indica anche l'insieme delle strutture e dei meccanismi inerenti la forma delle parole, inteso come attributo proprio di un sistema linguistico. La morfologia di una lingua è dunque l'insieme degli elementi, delle categorie e delle regole che riguardano la forma delle parole in quella lingua

Analizzando per successiva segmentazione un enunciato del linguaggio umano, l'unità minima *di prima articolazione*, ovvero la minima unità isolabile dotata di un significato proprio e preciso, è il *morfema*. Tuttavia tale significato non è autonomo rispetto a quello degli altri morfemi, ma è usato solo in una modalità di specificazione reciproca. Infatti il morfema può servire non solo a rappresentare il concetto di cui si sta parlando, ma anche e soprattutto a indicare se si tratta di un oggetto o di una azione, se l'oggetto è uno o sono molti, se l'azione è in atto o è conclusa, chi la compie, e così fino a determinare completamente il significato (e la forma) della parola.

Vediamo un esempio di come sia possibile scomponendo una parola individuare i morfemi costituenti. La parola *sgonfiavamo* è scomponibile nei morfemi *s-* (indicante negazione), *-gonfi-* (indicante immissione di aria in un corpo), *-av-* (indicante azione compiuta continuativamente nel passato), *-amo* (indicante un soggetto molteplice in cui è incluso il parlante). Ciascuno dei morfemi, dunque, rappresenta un significato specifico che modifica e determina il significato complessivo della parola di cui fa parte. La maggior parte delle parole è costituita in modo analogo a quello visto, ovvero con un nucleo semantico, o *radice*, cui vengono aggregate una o più *desinenze* ed eventualmente uno o più *affissi*, che modificano o determinano il significato fondamentale della radice. A seconda della categoria lessicale, la radice può configurarsi in un *tema*, ovvero in una forma derivata (ma non ancora una parola completa) che caratterizza tutte le forme della parola attestate per quella categoria lessicale. Ad esempio, temi derivati dalla radice *am-* sono *amor-*, *amabil-*, *amorevol-* e simili.

Metodi di analisi morfologica

Considerando i concetti appena introdotti, possiamo definire i due passi fondamentali dell'analisi morfologica. Il primo passo consiste nella scomposizione della frase in parole distinte, il secondo nella scomposizione delle singole parole in morfemi. A partire da questo schema generale, esistono due principali tecniche analisi morfologica nelle applicazioni di TE: lo *stemming* (basato prevalentemente su regole morfologiche) e l'analisi basata su dizionari lessicali.

Lo *stemming* (*estrazione del tema*) comprende un insieme di tecniche utilizzate per ricondurre le parole al loro tema originario. Si è visto nella breve introduzione alla morfologia che il tema è in generale diverso dalla radice ed

è una stringa comune ad un insieme di parole morfologicamente legate tra loro. Gli algoritmi di stemming compiono sostanzialmente due operazioni distinte: estrazione dei *suffissi* (affissi che seguono la radice) e registrazione. Il modulo estrattore di suffissi elimina da ciascuna parola una terminazione che ritiene essere il più lungo suffisso possibile. Inteso in tal senso, viene considerato suffisso una stringa che dal punto di vista linguistico è in realtà una concatenazione di più suffissi differenti. Come seconda operazione, i temi prodotti dall'estrattore vengono normalizzati da una funzione di registrazione per tener conto della presenza di eventuali variazioni grafiche.

Al contrario dello stemming, l'analisi basata su dizionari sfrutta informazioni puramente lessicali presenti in una apposita base di dati. Tra le varie tecniche comprese in questa classe ci limitiamo a citare il *word-based approach*, basato sul principio che nuove parole possono essere generate da vecchie mediante opportune regole di trasformazione e il *concatenative approach*, basato sulla conoscenza di tutte le forme base delle parole e di tutti i possibili suffissi flessivi che possono essere loro associati.

2.5.2 Analisi sintattica

Cenni di sintassi

Nell'ambito della linguistica, si chiama sintassi l'insieme delle norme che all'interno di una lingua regolano la formazione delle frasi e la disposizione degli elementi che le costituiscono. La struttura in cui si dispongono i significati all'interno dell'enunciato è chiamata *catena sintattica*. Questa struttura ha un impianto fortemente gerarchico dal punto di vista logico, indipendentemente da quale sia l'ordine della sequenza lineare in cui si susseguono gli elementi che la compongono.

Il *sintagma* è l'unità minima nella catena sintattica: costituisce cioè una stringa di parole accomunate dalla medesima funzione logica all'interno dell'enunciato. La parola fondamentale di un sintagma, senza la quale il sintagma non sussisterebbe o non sarebbe comprensibile, è chiamata *testa* del sintagma. Gli altri elementi, accessori più o meno importanti dal punto di vista del senso, ma sempre accomunati alla testa dalla funzione logica, sono detti *modificatori*.

I sintagmi vengono classificati a seconda della categoria grammaticale cui appartiene la parola che funge da testa. Avremo quindi sintagmi nominali, sintagmi verbali, sintagmi preposizionali, e così via. Tra due sintagmi può manifestarsi un rapporto di dipendenza: quando uno dei due non potrebbe sussistere senza l'altro si dice che dipende da questo. Tra i due, il sintagma indispensabile viene detto *dominante*. Generalmente si considera che la maggioranza delle frasi prodotte nella comunicazione umana sia composta almeno da un sintagma nominale e da un sintagma verbale, anche se questa regola non è di validità generale.

Per l'analisi della struttura di una frase è necessario ricorrere ad una rappresentazione dei livelli di dipendenza e dei legami logici tra i vari sintagmi. Il più diffuso schema di rappresentazione è lo *schema ad albero*, che analizza la frase a individuando a ciascun livello i sintagmi che lo popolano (nodi) e i rispettivi legami (rami) con i sintagmi a livello successivo, finché non si giunge alla descrizione completa della frase secondo i suoi costituenti atomici (foglie).

Analizzatori sintattici

Consideriamo qui in estrema sintesi soltanto alcune caratteristiche generali degli analizzatori sintattici, strumenti normalmente caratterizzati da un no-

tevole grado di complessità. Scopo di un analizzatore sintattico (*parser*) è quello di individuare la struttura sintattica di una frase, riconoscendone i sintagmi e le dipendenze logiche esistenti tra questi. Vi sono molti possibili algoritmi di parsing. Alcuni funzionano in modo *top-down*, iniziando dalla radice dell'albero ed espandendosi, seguendo le regole grammaticali, fino a combaciare con la stringa originaria costituente la frase. Altri usano una combinazione di *top-down* e *bottom-up*, altri ancora tecniche di programmazione dinamica volte ad evitare le inefficienze derivanti dal backtracking. Dal punto di vista storico, lo sviluppo delle tecniche di parsing è stato largamente influenzato dalle teorie linguistiche che si andavano via via sviluppando. Le prime tecniche di analisi sintattica (*deep parsing*), sulla scia degli studi di Chomsky, erano basate su grammatiche notevolmente sofisticate e miravano alla spiegazione esaustiva e particolareggiata della struttura della frase. Tali tecniche portavano ad algoritmi lenti ed inefficienti e furono gradualmente sostituite. Negli anni '80-'90, con la nascita della information extraction, si cominciò ad adottare modelli più semplici della struttura della frase, che portarono allo sviluppo delle attuali tecniche di *shallow parsing*, organizzate in un insieme stratificato di livelli di analisi. Citiamo tra questi il *tagging*, mirante ad etichettare le singole parole con la rispettiva categoria lessicale (nomi, verbi, aggettivi, ...) ed il *chunking*, consistente in un primo raggruppamento delle parole in piccoli insiemi contigui (chunks) di significato ben definito.

2.5.3 Costrutti linguistici ricorrenti

È stata più volte sottolineata l'importanza di individuare le relazioni concettuali esistenti tra i termini presenti nel corpus. Tali relazioni, si è anche detto, sono costituenti essenziali delle TKB e sono alla base della rete semantica

rappresentata sul piano linguistico nel corpus.

Vediamo qui ora come le informazioni inerenti le relazioni tra concetti possono rivelarsi estremamente preziose non solo al fine di rappresentare il dominio, ma anche come base partenza per l'estrazione di nuovi termini. La tecnica in questione si basa sull'osservazione che a particolari relazioni concettuali, come iperonimia, iponimia, meronimia e funzionalità corrispondono tipicamente costrutti linguistici (*linguistic patterns*) caratteristici e ricorrenti che nel linguaggio naturale sono per l'appunto utilizzati per esprimere quei particolari tipi di relazione. Dunque l'individuazione di certo pattern all'interno di un testo può essere indicativa della presenza di una specifica relazione concettuale, ed in questo caso gli elementi lessicali legati al pattern saranno con ogni probabilità dei buoni candidati termini. Sebbene tutti i linguaggi naturali presentino un comportamento generale di questo tipo, è opportuno che le tecniche che intendano giovare si concentrino non soltanto su una specifica lingua ma anche su un ben delimitato dominio. Le problematiche legate all'estrazione dei pattern linguistici saranno analizzate con maggior dettaglio in una apposita sezione, mentre conviene qui illustrarne ulteriormente la struttura generale.

I costrutti tipici ai quali siamo interessati sono essenzialmente di due classi, grammaticali e lessicali. Possono far parte della prima pattern che esprimono vincoli sulle parti del discorso coinvolte (per esempio NOME+VERBO) o sui legami sintattici che esse stabiliscono con il contesto locale. I pattern di tipo lessicale, invece, prevedono l'uso di opportune parole in presenza di particolari relazioni, come ad esempio "è parte di", "contiene", "è funzione di". Rivestono particolare importanza specifiche classi di verbi (verbi *definitivi*) particolarmente ricorrenti nella definizione di nuovi concetti. La ricerca di tali verbi in corpora di tipo manualistico/normativo può portare effettivamente

all'estrazione di numerosi termini essenziali all'interno del dominio.

2.5.4 Associazioni lessicali nei termini multi-word

Prendiamo in considerazione ora il compito specifico di individuare i termini di tipo complesso. Un aspetto importante di tali termini è la modalità di associazione delle parole che li compongono. Da un punto di vista probabilistico, i termini complessi sono insiemi di parole che tendono a ricorrere combinatamente nel corpus, veicolando un significato specifico che non è semplicemente la combinazione dei significati dei singoli costituenti. Tale fenomeno può essere quantitativamente misurato attraverso una ampia varietà di metriche, originariamente sviluppate in altri campi di ricerca. Metodi statistici, ad esempio, misurano la frequenza delle occorrenze combinate di più parole in una finestra fissata. Metodi presi in prestito dall'information retrieval includono invece funzioni di similitudine (Dice, Jaccard, coseno) applicate al modello a campo vettoriale già descritto in precedenza. Un metodo classico preso in prestito dalla teoria dell'informazione è la *Mutual Information* definita come

$$MI(X, Y) = \log_2 \frac{P(w_1, w_2)}{P(w_1)P(w_2)}$$

dove X e Y sono variabili aleatorie corrispondenti ai campioni considerati, in cui $x = w_1$ (prima parola) e $y = w_2$ (seconda parola) rispettivamente. La MI rappresenta il rapporto logaritmico della probabilità congiunta di trovare w_1 e w_2 nella finestra fissata sulla probabilità che tale evento avrebbe se le due variabili fossero indipendenti.

2.6 Probematiche di individuazione dei termini

2.6.1 Problemi di utilizzo dei costrutti ricorrenti

Sebbene la ricerca di pattern linguistici per l'acquisizione di conoscenza appaia una tecnica molto promettente, la sua applicazione concreta si presenta molto più difficoltosa del previsto. Innanzi tutto, ad un livello di organizzazione generale, l'indagine dei pattern presuppone una serie di precedenti fasi analitiche a più basso livello (parsing) non ancora in grado di garantire risultati sempre soddisfacenti.

Passando invece ai problemi propriamente inerenti i pattern, essi sono sostanzialmente distinguibili in due diverse classi: fanno parte della prima questioni relative alla ricchezza linguistica con cui essi si manifestano, mentre nella seconda troviamo ostacoli che impediscono a monte il loro uso in determinate circostanze.

Problemi insiti nella ricchezza linguistica

I pattern linguistici, siano essi di tipo lessicale o grammaticale, presentano in generale la caratteristica che, per ciascuna relazione concettuale, l'elenco dei pattern ad un certo momento individuati costituisce un insieme né piccolo né tantomeno stabile. Problemi di questo tipo includono:

Impredicibilità. Nella compilazione di liste di pattern sembra effettivamente che non si riesca mai a catturarli tutti. Mentre alcuni di essi sembrano abbastanza logici, come *è un* per l'iperonimia, e *contiene* per la meronimia, altri sembrano estremamente impredicibili. Si consideri l'esempio del verbo *colonizzare*. Fuori del contesto, non si sarebbe mai portati ad associare tale verbo con la meronimia. Tuttavia, nell'

esempio “*i microbi colonizzano il concime*” è chiaro come esso veicoli l’importante informazione che i microbi sono effettivamente costituenti del concime.

Polisemia. Un determinato pattern può essere portatore non solo di una particolare relazione concettuale, ma in contesti diversi anche di informazioni differenti. Si consideri ad esempio il verbo *definire*. Pur essendo esso estremamente significativo, in quanto utilizzato tipicamente per introdurre termini rilevanti, come in frasi del tipo “*X è definito come...*”, il medesimo verbo è spesso utilizzato nel senso di *creare*, come per esempio in “*definire le colonne*” in un word-processor. Altro esempio eloquente è dato dalla preposizione *di*. Sebbene essa indichi spesso meronimia, come in “*il concime è troppo povero di sali*”, la stessa può generare se utilizzata in questo senso anche un elevato rumore, come in “*concime di elevata umidità*”.

Dipendenza dal dominio Alcuni pattern sono chiaramente caratteristici di un dominio specifico. Ad esempio *specie* indica iperonimia in un contesto biologico (“*X è una specie di Y*”) esattamente come *sfumatura* la indica in un contesto cromatico (“*il lilla è una sfumatura del viola*”)

Limitazioni strutturali nell’uso dei pattern

In alcuni casi la conoscenza è espressa all’interno del corpus con mezzi strutturalmente diversi dai pattern linguistici. Si considerino alcuni esempi:

Uso di accorgimenti tipografici. In molti casi una relazione di iperonimia può essere espressa con l’uso delle parentesi, del trattino o anche della virgola (es: “*il concime, un composto nutritivo per il terreno*”).

Scrittura creativa. Quando un testo è scritto in un particolare stile artistico, diventa spesso arduo cercare pattern linguistici per l'acquisizione di conoscenza. Si pensi ad esempio a figure di senso come la metafora, la cui peculiarità è proprio quella di sostituire elementi lessicali con i loro corrispondenti di un campo semantico differente (es: *“Si è messa in moto la solidarietà”*)

Estensione del contesto. Tipicamente il campo di indagine nella ricerca di pattern linguistici è limitato ad una certa distanza dal candidato termine, ad esempio nell'ambito della medesima frase. Nasce quindi un serio problema ogni volta che il termine candidato non venga ripetuto nelle frasi successive, sia esso sottinteso o sostituito da un pronome.

Dalla sintesi dei vari possibili ostacoli brevemente richiamati, si coglie la necessità di ulteriori passi di ricerca prima che le tecniche di TE basate su pattern linguistici giungano ad un livello soddisfacente di precisione. Si manifesta in particolare la necessità di procedere all'analisi di grandi quantità di testi per arricchire la conoscenza specifica dei domini. Il vincolo ad uno specifico dominio applicativo sembra attualmente proporsi come uno dei problemi più difficili da superare.

2.6.2 Varianti nei termini complessi

Si è visto che la forma assunta da un termine presente in un testo può essere molto differente dalla sua forma paradigmatica utilizzata in una TKB o anche semplicemente in un dizionario. Venendo ora specificamente ai termini multiword, è intuibile come, a fronte di una struttura interna più complessa, la quantità di fattori potenzialmente portatori di varianti tenda naturalmente ad aumentare. Questo porta ovviamente con sé la necessità di maggiori sforzi

sul piano computazionale affinché tutte le varianti di ciascun termine siano correttamente riconosciute e riportate alla sua forma paradigmatica.

Quando è inserito in un contesto specifico, il termine deve soddisfare innanzitutto esigenze di comunicazione prima che eventuali requisiti formali. Tali esigenze sono generalmente individuabili in tre criteri: *economia*, *precisione*, *appropriatezza*.

Si consideri ad esempio il termine *olio di semi di girasole*. Questo, per il criterio di economia, può facilmente mutare nel più semplice *olio di girasole*. In direzione opposta, il criterio di precisione può determinare l'aggiunta di una o più specificazioni (*modificatori*) al nucleo originario del termine (*testa*). Come esempio di quest'ultimo caso si consideri la trasformazione da *amministratore delegato* a *amministratore delegato della società*.

Il criterio di appropriatezza agisce combinatamente ai due precedenti mettendo in gioco ulteriori esigenze derivanti dalla natura del contesto, dalla conoscenza del dominio, dal grado di generalità della frase, dal livello di formalità del documento.

2.7 Metriche di valutazione per la TE

Produrre criteri di valutazione significativi in ambito di TE è un problema fortemente dipendente dal tipo di applicazione che si intende analizzare. Un processo di acquisizione di conoscenza finalizzato alla costruzione di una TKB richiede evidentemente la definizione di attributi di qualità in grado di descrivere l'appropriatezza non soltanto di un semplice dizionario di termini ma di tutta la rete concettuale da questi indotta. Valutazioni di questo tipo sono in ultima analisi competenti ad esperti di dominio effettivamente qualificati nella valutazione della conoscenza rappresentata in quella specifi-

ca TKB. Con riferimento ad applicazioni meno complesse e pur tenendo ben ferme le distinzioni fatte in merito alla definizione di termine, presentiamo qui due metriche di valutazione tradizionalmente utilizzate nelle applicazioni di IR. L'obiettivo finale, in quel caso, è selezionare tutti e soli i documenti rilevanti fornendoli poi in risposta ad una certa interrogazione dell'utente (*query*). Dal punto di vista dell'indicizzazione automatica il processo può essere riletto in termini di estrazione di tutti e soli i "termini" notevoli (sono in realtà keywords!), comprendendo in questi le eventuali varianti.

Nel caso generale, le occorrenze estratte I_E si sovrappongono solo parzialmente con quelle effettivamente rilevanti I_R , inducendo nel processo una certa quantità di *rumore* (occorrenze irrilevanti erroneamente estratte) ed una certa quantità di *silenzio* (occorrenze rilevanti ma erroneamente non estratte). Questi due eventi sono alla base delle due tradizionali metriche di valutazione note come *precision* e *recall*:

$$P = \frac{|I_E \cap I_R|}{|I_E|}$$
$$R = \frac{|I_E \cap I_R|}{|I_R|}$$

Precision è la proporzione di oggetti rilevanti tra quelli estratti, mentre recall è la proporzione di oggetti estratti tra quelli rilevanti. Elevati valori di precision (prossimi ad 1) indicano basso rumore, mentre valori elevati di recall indicano basso silenzio. Queste due misure tuttavia non definiscono completamente il quadro in quanto il problema in esame è caratterizzato effettivamente da tre gradi di libertà. La metrica utilizzata generalmente per completare la descrizione è il *fallout*, o il suo complementare *precision of fallout*:

$$\Pi = \frac{\|(I \setminus I_R) \setminus I_E\|}{\|I \setminus I_R\|}$$

corrispondente alla proporzione di oggetti non rilevanti sui non estratti, dove I è l'insieme di tutti gli oggetti considerati.

Capitolo 3

Apprendimento automatico

Nel precedente capitolo sono stati presentati i concetti di base ed alcune problematiche generali relative alle applicazioni di estrazione terminologica. In particolare è stata presa in considerazione la possibilità di sfruttare nell'individuazione dei termini la loro informazione contestuale. Nei successivi capitoli verrà descritta una piattaforma sperimentale costruita a supporto di tali ipotesi. Le tecniche modellistiche ed algoritmiche di cui si farà uso discendono dalla teoria dell'Apprendimento Automatico (*machine learning*), i cui tratti essenziali sono oggetto del presente capitolo. Dopo una esposizione generale delle principali questioni inerenti la materia, sarà dato particolare rilievo ai problemi di classificazione e agli alberi di decisione.

3.1 Il processo di apprendimento

La capacità di apprendere è universalmente riconosciuta come caratteristica essenziale di qualsivoglia sistema supposto "intelligente". Lo studio dei processi alla base dell'apprendimento è quindi per sua stessa natura multidisciplinare e costituisce un punto di confronto per una grande varietà di

settori scientifici apparentemente non collegati, come l'intelligenza artificiale e la psicologia cognitiva. In particolare, l'*apprendimento automatico* si occupa dello studio dei processi computazionali alla base dell'apprendimento stesso, tanto negli elaboratori artificiali quanto negli stessi esseri umani che li programmano. È utile sottolineare che se da un lato l'aspetto "computazionale" può far sembrare artificioso questo parallelo, dall'altro la complessità degli algoritmi di machine learning raggiunge facilmente un livello al quale non si può prescindere da problematiche che associamo molto più comunemente all'essere umano che non ad una macchina di calcolo, come la natura dei concetti e delle relazioni che li legano, il ragionamento per induzione, le reazioni a fronte di stimoli esterni. Accade quindi che gli studi psicologico-cognitivi sull'apprendimento e quelli specifici dell'intelligenza artificiale tendano a completarsi reciprocamente: mentre quest'ultima ha la possibilità di sviluppare teorie ed applicazioni a partire da modelli non suoi (caso emblematico le reti neurali), i primi possono trovare nelle tecniche specifiche dell'apprendimento automatico preziosi strumenti di verifica.

3.2 Il contesto ambientale

Si è fatto solo un rapido cenno alla grande varietà di questioni legate al concetto di apprendimento, ma dovrebbe risultare abbastanza chiara la grande difficoltà di darne una definizione allo stesso tempo concisa ed esaustiva. Ponendoci tuttavia in un quadro di riferimento più orientato alle applicazioni, potremmo tentare empiricamente di definire l'apprendimento come il *miglioramento delle prestazioni in un determinato ambiente, ottenuto con una progressiva acquisizione di conoscenza risultante da esperienza maturata internamente all'ambiente stesso*. La caratteristica più importante di questa

definizione è la sua affermazione che l'apprendimento non può essere descritto come processo a se stante. L'agente che apprende è collocato in un *ambiente* definito ed è su tale ambiente che può eventualmente sperare di apprendere alcunché. Sempre dalla definizione si evince che l'agente è inoltre legato ad un certo patrimonio di conoscenza dal quale può attingere o al quale può aggiungere nuovi elementi. *Ambiente e conoscenza* sono dunque due componenti chiave, dalla cui definizione dipende imprescindibilmente la qualità se non addirittura la possibilità stessa del processo di apprendimento.

La tipologia degli elementi portatori di conoscenza reperibili in un dato ambiente determina direttamente una importante caratterizzazione del processo stesso di apprendimento. Questo è detto *supervisionato* o *non supervisionato* a seconda che l'agente possa o meno ottenere un feedback relativamente all'efficacia delle azioni compiute.

Si consideri l'esempio di un sistema automatico di frenata in un veicolo. Tale sistema si troverà ad operare in una grande varietà di condizioni atmosferiche e di fondo stradale. Supponiamo che la conoscenza a priori definita dal progettista non sia sufficiente a descrivere tutte le possibili condizioni operative, ci saranno dunque casi in cui il sistema, acquisito tramite sensori un certo numero di elementi esterni (velocità, temperatura, umidità, caratteristiche del fondo stradale) si troverà a dover “tentare” una certa (re)azione, consistente nella quantità di potenza erogata nell'azione di frenata. Una volta fermato il veicolo, note le condizioni iniziali descritte e l'azione intrapresa in condizioni di parziale incertezza, risulta estremamente prezioso per il sistema “intelligente” conoscere l'effettivo spazio di frenata che è stato in grado di offrire come risultato della sua azione.

Tale informazione, confrontata con altre ottenute in circostanze analoghe, può permettere al sistema di ragionare in modo induttivo ed apprendere

quale sia effettivamente la forza frenante più opportuna al ripetersi di quelle determinate circostanze esterne. Con l'aumentare dell'esperienza ed il procedere dell'apprendimento risulteranno via via sempre più ridotti i casi in cui il sistema è costretto a prendere decisioni in condizioni di incertezza.

Quello descritto ora è un caso tipico di apprendimento supervisionato, in cui cioè l'ambiente rende disponibili informazioni tanto sul suo stato attuale quanto sull'esito delle azioni intraprese. E' naturalmente intuibile come gli ambienti supervisionati siano estremamente più favorevoli dei non supervisionati, mentre non stupisce il fatto che questi ultimi siano di fatto la regola più che l'eccezione. Spesso infatti la "supervisione" è artificialmente creata in condizioni di laboratorio tramite la presenza di un "istruttore", che interviene in quella fase della "vita" dell'agente in cui esso riceve quella conoscenza *a priori* che si rivela estremamente preziosa soprattutto in ambienti "ostili".

3.3 La rappresentazione dell'ambiente

Accanto al grado di supervisione, altre caratteristiche contribuiscono a rendere un ambiente più o meno "accogliente" dal punto di vista dell'apprendimento. La sua complessità, ad esempio, è una caratteristica essenziale, nella misura in cui si ripercuote nella difficoltà di descrivere i concetti. Concetti intrinsecamente più sofisticati sono più difficili da apprendere, come accade ad esempio con le grammatiche nei domini linguistici.

La complessità dell'ambiente appare spesso in relazione ad un'altra caratteristica negativa: la quantità di attributi (features) *irrilevanti* ai fini dell'apprendimento. Spesso non è opportuno o non è possibile selezionare a priori le features su cui condurre il processo, ed un numero troppo elevato di informazioni irrilevanti può costituire un grosso ostacolo al sistema che

cerca di isolarle e scartarle. Altro fattore limitante è il rumore. Nei sistemi di apprendimento questo può riguardare tanto la descrizione dell'oggetto (le features) quanto il feedback che il sistema riceve dall'ambiente. Il problema dell'eliminazione del rumore è aggravato dal fatto che, essendo tipicamente l'ambiente per lo più incognito, oggetti nuovi e lievemente differenti da quelli noti rischiano di essere erroneamente classificati come rumorosi.

Altri fenomeni di disturbo possono essere legati non tanto all'ambiente ma al linguaggio adottato per descriverne il modello che verrà effettivamente sottoposto all'agente. Il problema della modellizzazione e rappresentazione dell'ambiente costituisce un punto particolarmente delicato, in quanto il modello può implicitamente nascondere determinate features rilevanti ed esaltarne viceversa altre del tutto inutili. In ambienti complessi, la modalità scelta per la loro rappresentazione può condizionare notevolmente il risultato del processo.

Il dominio del linguaggio naturale, nel quale questo lavoro si pone, soffre in modo marcato di tali problematiche ed ha richiesto un ulteriore sforzo in fase sperimentale, valutando contemporaneamente più modelli in parallelo, come verrà descritto nel seguito.

3.4 Apprendimento nei problemi di classificazione

I problemi di classificazione costituiscono un dominio frequentemente associato a processi di apprendimento. In tali problemi non si tratta di intraprendere azioni ma di classificare oggetti. Il presente lavoro si colloca esso stesso in un dominio di classificazione, in quanto l'estrazione di terminologia è una operazione che presuppone per definizione il riconoscimento di un *termine* da un *non-termine*. E' opportuno dunque rivisitare i concetti sin qui

esposti in materia di apprendimento automatico nel contesto più specifico dei problemi di classificazione. Ciò che abbiamo sin qui definito “agente” è dato ora da una pura funzione di decisione. L’ambiente è costituito da oggetti tutti dello stesso tipo, descritti cioè da un comune insieme di features selezionate a priori. Queste sono raccolte in un vettore (feature vector) e ciascun oggetto specifico costituisce una sua istanza (o *esempio*, o *campione*). Inoltre l’insieme di tutti gli oggetti disponibili è partizionato in modo tale che ciascuno appartenga ad una ed una sola classe. Compito della funzione di decisione è quindi quello di analizzare gli oggetti (esempi) che le vengono sottoposti, valutarne le features, e stabilirne l’appartenenza ad una classe piuttosto che alle altre.

Abbiamo quindi che, nel contesto di un problema di classificazione, l’oggetto del processo di apprendimento è proprio la funzione di classificazione. L’efficacia dell’apprendimento sarà allora quantificabile con l’efficienza della funzione di decisione valutata nel suo classificare gli oggetti. La funzione di decisione viene appresa per induzione da un algoritmo opportuno, al quale vengono sottoposti un certo numero di campioni insieme alle relative classi di appartenenza. In questo caso specifico, cioè di *apprendimento supervisionato in problemi di classificazione*, il processo è talvolta detto “learning by examples” Viceversa, nel caso di apprendimento non supervisionato, *non* viene fornita all’algoritmo di apprendimento la classe dei campioni sottoposti e questo opera alla cieca, basandosi esclusivamente su dati relativi all’efficienza a posteriori della funzione, per la cui costruzione viene utilizzato un approccio incrementale.

Un processo di apprendimento non supervisionato in problemi di classificazione può essere motivato sia dalla non conoscenza previa delle classi *neanche* da parte dello sperimentatore (ed in questo caso lo scopo del pro-

cesso è proprio quello di scoprirle) sia dalla necessità di una verifica indiretta della coerenza delle classi stesse (questa volta note) con gli oggetti disponibili.

3.5 Alberi di decisione

Gli alberi di decisione costituiscono una delle possibili modalità di rappresentazione della funzione di decisione. Essi soffrono di alcune limitazioni riguardo la classe di funzioni esprimibile ma offrono il vantaggio di una struttura relativamente semplice, motivo per cui sono stati spesso adottati come base di partenza per lo sviluppo di algoritmi di apprendimento di notevole efficacia, di cui alcune applicazioni degne di nota saranno citate nel seguito.

La struttura generale di un albero di decisione rispecchia intuitivamente quello che potremmo definire un comportamento “razionale” in un agente umano impegnato in un problema di classificazione. Infatti, percorrendo un albero di decisione dalla radice verso le foglie, il processo decisionale prende in considerazione le features dell’oggetto da classificare in un modo preordinato, partendo cioè da quelle più rilevanti (meglio, più *discriminanti*) fino a quelle più particolari o secondarie. Data quindi la descrizione (feature vector) della classe di oggetti da trattare, i nodi interni dell’albero sono un sottoinsieme delle features disponibili (eventualmente tutte) mentre i rami corrispondono ad un sottoinsieme delle loro possibili istanze. Ciascun nodo foglia, infine, corrisponde ad una possibile classificazione dell’oggetto esaminato.

Appare dunque chiaro che un albero di decisione, una volta disponibile, costituisce una funzione di decisione di pronta applicazione in un dominio ben definito. È viceversa compito ben più articolato la generazione (dunque l’*apprendimento*) dell’albero stesso prima che possa essere utilizzato, problema di cui ora ci occuperemo. Come già affermato, ci siamo posti in un particolare campo di azione (problemi di classificazione in ambiente supervi-

sionato) in cui il metodo di apprendimento è spiccatamente induttivo ed il processo relativo è detto *learning by examples*. Si tratta infatti di costruire un algoritmo che generi un albero di decisione a partire da un certo numero di osservazioni (*examples*) che gli vengono opportunamente fornite da un istruttore. Occorre ancora sottolineare che, essendoci posti in un ambiente supervisionato, vengono fornite all'algoritmo tanto le descrizioni degli oggetti prescelti (dunque istanze di feature vector) quanto le rispettive classificazioni degli oggetti stessi. Partendo da un numero sufficiente di "buone" osservazioni, un algoritmo di apprendimento dovrebbe essere in grado di generare un albero di decisione allo stesso tempo *sintetico* ed *efficace*.

Vediamo con maggiore dettaglio in cosa consistono queste due caratteristiche. L'efficacia dell'albero è direttamente associata alla sua capacità di classificare correttamente oggetti *non visti* in precedenza (cioè nella fase di apprendimento). Per tale ragione un corretto processo di apprendimento di alberi di decisione presuppone la definizione previa di tre insiemi di oggetti: l'*insieme di apprendimento* (*learning set*) è costituito di tutti gli oggetti disponibili di cui si intende far uso. Tale insieme viene quindi suddiviso in un *insieme di addestramento* (*training set*) e in un *insieme di valutazione* (*test set*).

Partizionati in questo modo gli oggetti a disposizione, l'albero di decisione viene appreso esclusivamente a partire dal training set e viene valutato esclusivamente sul test set. Si noti che una ulteriore valutazione dell'albero, basata questa volta sullo stesso learning set usato per la sua generazione, è una operazione di significato non banale e di esito niente affatto scontato. La spiegazione di tale affermazione ci spinge direttamente verso il secondo attributo di qualità introdotto, ovvero la *sintesi*.

Si consideri un ipotetico algoritmo di apprendimento che si limiti a me-

morizzare tutti gli esempi del training set che gli vengono sottoposti in fase di apprendimento. Riprendiamo ora l'ipotetico esperimento proposto poco fa: l'albero generato da tale algoritmo sarebbe in grado di riconoscere uno per uno tutti gli esempi contenuti nel training set, qualora si decidesse di valutarli. Tuttavia la capacità predittiva di tale albero, cioè la sua *efficacia*, risulterebbe praticamente nulla qualora gli fossero sottoposti gli esempi (non visti) contenuti nel test set. La caratteristica essenziale di un buon algoritmo di apprendimento dovrà allora essere quella di saper ipotizzare per induzione quali tra le features degli oggetti in esame risulteranno maggiormente significative ai fini della classificazione. Tali features verranno non solo certamente prese in considerazione, ma anche poste in prossimità della radice dell'albero affinché vengano valutate per prime in fase di decisione. Viceversa, le features meno rilevanti verranno relegate in posizioni più periferiche dell'albero o addirittura scartate in quanto ritenute non significative. Al contrario del precedente, un albero generato con tali criteri non è più un semplice elenco di osservazioni, ma una descrizione sintetica ed efficace del training set. A questo punto, se la suddivisione tra training set e test set è stata fatta correttamente (cioè in modo casuale), è probabile che non tutti gli oggetti del training set vengano nuovamente riconosciuti, ma con ogni probabilità l'albero avrà buone caratteristiche di predittività sul test set, cioè buona efficacia. Con questo abbiamo sostanzialmente isolato il problema principale dell'apprendimento di alberi: la selezione delle features.

Il criterio per valutare la maggiore o minore rilevanza di una feature è il suo *grado di discriminazione* rispetto alle classi degli esempi di addestramento. In altre parole, scelta una feature da valutare (che supponiamo di tipo discreto o enumerativo), si elencano tutti i suoi valori osservati nel training set. Per ciascuno di tali valori, si considerano tutti gli oggetti che lo assu-

mono per quella feature e se ne controllano le classi di appartenenza. Nel caso migliore, per ciascuno dei valori della feature, tutti gli oggetti tenderanno a disporsi nella medesima classe. In questo caso la feature selezionata sarà *totalmente discriminante*, in quanto per ciascuno dei suoi valori sarà univocamente ipotizzabile la classe dell'oggetto esaminato, ed a ciascuno di tali valori corrisponderà un nodo foglia dell'albero. Nel caso in cui per uno o più valori della feature la classe di appartenenza non sia univoca, occorre una ulteriore indagine, dunque si procede valutando una feature ulteriore tra quelle non ancora considerate, ed aggiungendo all'albero un ulteriore nodo interno.

Illustriamo ora un esempio. Vogliamo costruire un albero di decisione in grado di valutare se una giornata è buona o meno per giocare a golf. Si tratta di un problema di decisione con due possibili *classi* (Play, Don't play) e quattro features osservabili, per ciascuna delle quali sono indicati i possibili valori. Questi possono appartenere ad un insieme discreto o ad un range numerico continuo.

outlook: sunny, overcast, rain.

temperature: continuous.

humidity: continuous.

windy: true, false.

Vediamo ora un possibile training set, costituito da 14 istanze differenti. Si tratta verosimilmente di un insieme di situazioni reali in cui, date le condizioni atmosferiche, un agente umano deciderebbe di giocare o meno una partita di golf.

sunny, 85, 85, false, Don't Play

sunny, 80, 90, true, Don't Play

```
overcast, 83, 78, false, Play
rain, 70, 96, false, Play
rain, 68, 80, false, Play
rain, 65, 70, true, Don't Play
overcast, 64, 65, true, Play
sunny, 72, 95, false, Don't Play
sunny, 69, 70, false, Play
rain, 75, 80, false, Play
sunny, 75, 70, true, Play
overcast, 72, 90, true, Play
overcast, 81, 75, false, Play
rain, 71, 80, true, Don't Play
```

Un possibile albero di decisione apprendibile a partire da questi dati è il seguente:

```
outlook = overcast: Play (4.0)
outlook = sunny:
|  humidity <= 75 : Play (2.0)
|  humidity > 75 : Don't Play (3.0)
outlook = rain:
|  windy = true: Don't Play (2.0)
|  windy = false: Play (3.0)
```

L'algoritmo ha selezionato *outlook* come feature maggiormente discriminante. Infatti nel caso di *outlook=overcast*, in ben 4 casi su 14 (28.6%) si deciderebbe di giocare, mentre per gli altri due valori (*sunny* e *rain*) sarebbe opportuno considerare altri elementi (rispettivamente *humidity* e *windy*). Si noti come nel caso di *humidity*, dichiarata di tipo continuo, l'algoritmo sia

stato in grado di individuare una soglia discriminante (75% di umidità). A sostegno della precedente discussione sull'apprendimento dell'albero, è bene sottolineare espressamente che, sebbene questo specifico albero classifichi correttamente *tutti* i 14 esempi del training set, è stata intenzionalmente trascurata nell'apprendimento la feature *temperature* in quanto ritenuta poco discriminante o, nel caso specifico, addirittura ridondante.

3.6 Il sistema di apprendimento C4.5

Gli alberi di decisione presentati (precisamente *univariate decision trees*) costituiscono un importante caso speciale delle gerarchie concettuali (concept hierarchies). Si tratta di un campo di ricerca nel quale l'applicazione di concetti propri di altre discipline (statistica, teoria dell'informazione, calcolo delle probabilità, etc.) ha contribuito allo sviluppo di varie generazioni di algoritmi basati su tecniche alternative o complementari tra loro ed applicabili a fasi differenti della costruzione degli alberi. Il successo di alcuni algoritmi di apprendimento basati su tali tecniche ne ha suggerito addirittura lo sfruttamento commerciale. C4.5 è il sistema di apprendimento automatico utilizzato nel presente studio. Esso, sviluppato da Ross Quinlan, ha dato più di una volta prova di grande efficacia e flessibilità, specialmente in casi dove l'applicazione di metodi tradizionali basati sulla costruzione di modelli non era effettuabile per la grande complessità dei fenomeni in esame, come ad esempio il comportamento dei corpi elettorali.

Una applicazione sorprendente di C4.5 risale al 1992, quando un sistema basato su alberi di decisione fu inserito nel ciclo di controllo del simulatore di volo di un Cessna. L'insieme di apprendimento fu generato dall'osservazione di tre piloti che eseguivano per trenta volte ciascuno un determinato piano

di volo. Si fece uso di 20 variabili di stato descrittive lo stato del velivolo e dell'ambiente circostante. Il risultato fu eccellente, in quanto il controllore automatico si mostrò in grado di generalizzare il comportamento dei suoi insegnanti filtrando gli errori occasionali compiuti da ciascuno.

C4.5 è una versione precedente del sistema effettivamente commercializzato e si tratta di un tool di apprendimento flessibile ed efficiente i cui dettagli di configurazione ed uso sono stati raccolti in appendice.

Capitolo 4

Apprendimento automatico di terminologia

Nei precedenti capitoli è stato condotto un excursus di argomenti pertinenti a settori dell'intelligenza artificiale particolarmente differenti. Nell'ambito dell'elaborazione del linguaggio naturale, sono stati trattati i concetti di base della terminology extraction, le sue finalità ed alcune tecniche applicative di maggior uso. Nell'ambito dell'apprendimento automatico, il secondo dei due grandi settori presentati, abbiamo trattato alcuni concetti di riferimento generali per poi concentrarci in modo particolare su classe di applicazioni ben definita: i problemi di classificazione basati su alberi di decisione in ambito supervisionato. Scopo del presente capitolo è di operare finalmente una sintesi, presentando il nostro particolare approccio alla terminologia di dominio.

4.1 Obiettivo specifico di questo studio

Acquisiti i concetti e gli strumenti di base, possiamo ora procedere alla presentazione dell'argomento specifico su cui verte questo studio: l'applicazione del machine learning alla terminology extraction. Più in particolare, si vuole tentare sperimentalmente l'applicazione degli alberi di decisione alla indi-

viduazione di termini un un corpus con l'utilizzo esclusivo di informazione esogena. In altri termini, la questione che ci si pone è in quale misura l'informazione contestuale relativa ad un candidato termine sia utilizzabile in termini operativi per stabilirne in qualche modo la "termohood" (o "termship") cioè l'effettiva appropriatezza in un fissato dominio. È opportuno sottolineare che l'indagine condotta mirerà a valutare l'informazione contestuale *in modo esclusivo*, sebbene i migliori risultati nelle applicazioni reali siano normalmente ottenuti congiuntamente alle specifiche tecniche di ricerca analizzate in precedenza, cioè quelle legate alla struttura propria del termine assoluto (informazione endogena).

Ci apprestiamo quindi all'uso concreto degli alberi di decisione, in particolare facendo uso del già descritto sistema di apprendimento C4.5. Vale la pena richiamare brevemente che essi costituiscono uno strumento di rappresentazione intuitivo e maneggevole ma intrinsecamente rigido dal punto di vista modellistico. Si aprono in particolare due problemi principali che tratteremo nei prossimi paragrafi: il primo, più generale, relativo ad un importante requisito dell'insieme di apprendimento, ed il secondo più specificamente modellistico, relativo alla rappresentazione dell'informazione esogena che si vuole sfruttare.

4.2 Il problema dei *non-termini*

Abbiamo precedentemente annunciato l'intenzione di trattare il problema della terminologia in un contesto squisitamente supervisionato. Come ampiamente descritto nella parte relativa all'apprendimento automatico, tale approccio comporta il dover fornire all'algoritmo di apprendimento un opportuno insieme di addestramento costituito di campioni *completamente descrit-*

ti, vale a dire oggetti di cui siamo tenuti a dichiarare tanto una descrizione completa (tramite il feature vector) quanto la reale *classe di appartenenza*.

Venendo al contesto terminologico, essendo noi interessati in ultima analisi a separare tutti i candidati termini presenti nel corpus in *termini* effettivi e *non-termini*, quanto appena detto sull'insieme di addestramento comporta la necessità di un insieme di non-termini su cui basare l'apprendimento. È bene chiarire che la questione non è in alcun modo eludibile: si osservi infatti che in mancanza di “campioni negativi” il sistema è indotto ad apprendere un albero di decisione banale, composto cioè di un unico nodo foglia contenente l'unica classe di termini osservata, cioè quella positiva. La questione dei campioni negativi è dunque un problema, a monte della fase di apprendimento, di pertinenza specifica del modello esteso di dominio in cui ci siamo posti. Si era già fatto riferimento alla necessità, accanto al corpus stesso, di un dizionario controllato di termini, un *oracolo* su cui basarsi. Nel nostro caso specifico, è stato fatto uso di un tale dizionario per operare una precisa suddivisione in positivi e negativi su tutti i candidati termini estratti dal corpus. Come sarà spiegato con maggior dettaglio, tale partizionamento è stato utilizzato per verificare a posteriori l'effettiva rilevanza dei termini negativi nella fase di apprendimento.

4.3 Rappresentazione dell'informazione contestuale

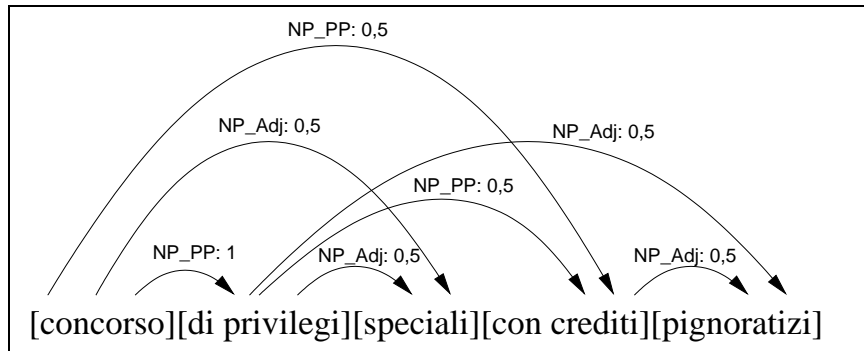
Abbiamo detto che l'uso degli alberi di decisione nella fase di apprendimento impone la definizione di un preciso modello di rappresentazione. Tale modello deve essere espresso, in ultima analisi, con un unico vettore individuato da un fissato numero e tipo di features. Ora, i dati sull'informazione contestuale relativa ai candidati termini vengono estratti dal corpus

di riferimento con l'ausilio di uno specifico analizzatore sintattico chiamato *CHAOS*. Anticipiamo dunque che saremo interessati ad una conversione tra i dati forniti da *CHAOS* ed un modello basato su feature vector su cui operare l'apprendimento. Ma occupiamoci ora del particolare modello usato da *CHAOS*.

Una caratteristica peculiare di questo strumento, la sua modularità, interessa direttamente il nostro problema modellistico. L'approccio modulare alla sua creazione ha permesso infatti lo sviluppo di componenti funzionalmente distinte e componibili in vario modo, con lo scopo di poter soddisfare con flessibilità esigenze mutevoli di velocità e precisione. Questa libertà di configurazione ha comportato la definizione di uno specifico modello strutturale denominato *eXtended Data Graph (XDG)*. Si tratta sostanzialmente di un grafo rappresentante le unità sintattiche (*chunks*) individuate all'interno della frase, poste in corrispondenza dei nodi, e le relative dipendenze (o relazioni) presenti tra questi, corrispondenti agli archi.

Diverse sono le caratteristiche rilevanti presenti in un *XDG*. Innanzitutto esso permette la rappresentazione delle forme di ambiguità che i moduli di analisi sintattica non sono riusciti ad eliminare. Tali ambiguità sono rappresentate da una molteplicità di archi nell'ambito di una medesima relazione. Il formalismo *XDG* permette inoltre la rappresentazione tanto dell'informazione endogena che di quella esogena ed ammettendo la coesistenza di più interpretazioni possibili. Nel caso dei termini complessi, infatti, le dipendenze sintattiche tra i vari chunk (*inter chunk dependencies, icds*) assumono il ruolo di informazione endogena oppure esogena in relazione alle ipotesi fatte sulla struttura del candidato termine. Si consideri ad esempio il sintagma nominale "*Concorso di privilegi speciali con crediti pignoratizi*", il cui

Figura 4.1: Esempio di grafo XDG



corrispondente grafo XDG è riportato in figura.

I diversi chunk individuati in fase di analisi sono racchiusi tra parentesi quadre. Gli archi rappresentano le relazioni esistenti tra questi e sono etichettati con le categorie sintattiche relative all'elemento di partenza e a quello di arrivo. Nell'ambito di ciascuna relazione, l'arco è da considerarsi uscente dal termine dominante ed entrante su quello "catturato". Il numero presente su ciascun arco indica inoltre (se minore di 1) l'ambiguità persistente, relativa a quella relazione, che il parser non è riuscito ad eliminare. Consideriamo ad esempio il chunk preposizionale *con crediti*. Esso ha due archi entranti in quanto non è possibile stabilire a priori se sia catturato da *concorso* o da *privilegi*. La plausibilità totale della relazione (1 = certezza) è dunque suddivisa in parti uguali (=0.5) tra i suoi due archi entranti in *con crediti*.

Dal punto di vista dell'informazione endogena ed esogena, il grafo XDG non opera alcuna selezione previa, in quanto esso non contiene ipotesi sui possibili candidati termini. Tale informazione viene quindi "proiettata" sul grafo nel momento in cui tali ipotesi vengono formulate. Se per esempio considerassimo *privilegi speciali* come candidato termine, la relazione presente tra *privilegi* e *speciali* verrebbe a trovarsi compresa all'interno del termine

stesso, ed assumerebbe il ruolo di informazione esogena. Viceversa, mantenendo la medesima ipotesi, l'arco entrante su *privilegi* relativo alla relazione con *concorso* sarebbe un elemento di informazione esogena, in quanto la parola *concorso* non fa parte del candidato termine prescelto ed è quindi relativa al suo contesto sintattico. Scopo degli esperimenti che verranno descritti nel seguito è precisamente quello di catturare tutte e sole le relazioni di quest'ultimo tipo e di verificare se esse possono essere raccolte in un modello significativo ai fini della distinzione tra termini e non-termini.

4.4 Il problema della modellizzazione

Il formalismo grafico fin qui descritto costituisce ora il punto di partenza su cui sviluppare il processo di apprendimento. È ancora il caso di richiamare le osservazioni fatte nella trattazione del machine learning relativamente all'importanza della rappresentazione dell'informazione e della reale dipendenza da essa di tutto l'esito dell'apprendimento. Il fatto di rappresentare determinati aspetti e non altri dell'ambiente, oppure una diversa modalità di interpretazione di certe caratteristiche è un problema che ora ci troviamo a dover concretamente affrontare nell'elaborazione di un modello. Si tratta dunque di estrarre l'informazione contestuale, *e solo quella*, presente nell'XDG e rappresentarla attraverso un insieme di feature. Il problema presenta decisamente una notevole quantità di possibili soluzioni. Alcune di queste sono state considerate, realizzate sperimentalmente e confrontate, come sarà diffusamente descritto nel prossimo capitolo.

Capitolo 5

Modelli di informazione contestuale

Nei precedenti capitoli sono stati diffusamente presentati i presupposti teorici e gli strumenti di rappresentazione che andremo ad utilizzare per le nostre applicazioni sperimentali. L'unico elemento attualmente mancante, per poter applicare gli alberi di decisione all'informazione contestuale offerta dal corpus, è ora un meccanismo di traduzione da eXtended Data Graph a feature vector. Le modalità con cui effettuare il passaggio sono molteplici e fanno riferimento a caratteristiche di *tipologia* e di *ambito* dell'informazione contestuale da rappresentare. Rispetto alla tipologia, si può decidere di utilizzare solo informazione di tipo *sintattico* oppure solo di tipo *lessicale*, ovvero una combinazione di entrambe. Rispetto invece all'ambito, si può decidere di operare *localmente*, con riferimento, cioè, alle singole occorrenze dei candidati termini, oppure *globalmente*, cioè con riferimento al comportamento "aggregato" di tutte le occorrenze di un medesimo candidato termine (più precisamente della medesima *testa*) nell'ambito dell'intero corpus.

Fissati gli opportuni valori dei parametri appena descritti, si ottiene un modello ben definito di informazione contestuale. I dati iniziali contenuti nell'XDG possono così essere trasformati nel nuovo modello e forniti diretta-

mente al sistema C4.5. Nell'ambito del presente studio sono stati realizzati quattro di tali modelli, che saranno presentati nel corso del capitolo. Si noti, inoltre, che la scelta di non utilizzare *alcuna* tipologia di informazione contestuale è anch'essa significativa e conduce direttamente ad un particolare modello di rappresentazione, quello relativo alla *frequenza* pura.

5.1 Formato dei dati disponibili

I dati estratti dal corpus tramite CHAOS e codificati nel grafo XDG sono, in realtà, memorizzati su un file dal medesimo contenuto informativo del grafo ma di formato proprio. Per maggiore precisione, i dati vengono distribuiti in *due* diversi files a seconda della loro appartenenza all'insieme degli esempi positivi o negativi. Come precedentemente discusso, tale distinzione è fornita a priori da un oracolo e riveste un ruolo essenziale dal momento che, nel nostro problema, non è possibile prescindere da una classificazione previa degli oggetti. Con riferimento al precedente esempio di XDG, relativo al sintagma “*Concorso di privilegi speciali con crediti pignoratizi*”, il corrispondente segmento di dati fornito dall'analizzatore (trascurando per il momento la classe positiva o negativa) è il seguente:

concorso	NORel nohead	1
privilegi	NP_PP concorso	1
crediti	NP_PP concorso	0.5
crediti	NP_PP privilegi	0.5

In questo segmento ciascuna riga corrisponde ad una relazione, avente il grado di plausibilità, indicato tra la *testa* del termine (nel primo campo) e la parola con la quale l'analizzatore ritiene sia in relazione (indicata nel terzo campo).

Si noti come *speciali* e *pignoratizi* siano stati intenzionalmente omessi in quanto non rispondenti ai vincoli formali richiesti per una testa. Si tratta in questo caso di aggettivi il cui ruolo è tipicamente quello di modificatore. Si noti, però, come questi non siano stati presi in considerazione *neanche* in tale qualità, poiché un'eventuale relazione testa-modificatore, nell'ambito del medesimo termine (così come quella tra *privilegi* e *speciali*), costituisce in realtà un elemento di informazione endogena che, in quanto tale, viene esclusa dalla nostra analisi.

5.2 Formato richiesto da C4.5

Il sistema di apprendimento C4.5 richiede il rispetto di un preciso formato di rappresentazione dei dati in ingresso. Le informazioni necessarie sono memorizzate in due diversi files: il primo (*.names) contenente la descrizione delle features e delle classi di appartenenza, il secondo (*.data) contenente il learning set vero e proprio. Volendo richiamare i concetti introdotti nella trattazione degli alberi di decisione, si ricorda che gli elementi del learning set altro non sono che *istanze* di feature vector. Secondo tale schema, il file .names contiene la descrizione del feature vector (sebbene le componenti possano essere di tipi differenti), mentre il .data contiene come righe i vettori numerici veri e propri relativi ai singoli elementi. Un esempio di tali files è stato dato precedentemente nell'esempio relativo alla partita di golf. Lo scopo del software realizzato, che verrà descritto più avanti, è innanzitutto quello di effettuare una traduzione dal formato dei dati disponibili a quello richiesto da C4.5, secondo il modello di rappresentazione scelto.

5.3 Modello locale-lessicale (L)

Caratteristica essenziale di questo modello è l'utilizzo di informazioni reperite nel corpus esclusivamente nel contesto *locale* di ciascun termine, o più precisamente di ciascuna sua occorrenza. L'informazione rappresentata è, fondamentalmente, di tipo lessicale, anche se viene in realtà classificata per categorie sintattiche. Il "prototipo" di feature vector è dato infatti dalla successione di tutti i tipi di relazione presenti nel corpus, mentre ciascuna istanza, cioè ciascuna occorrenza di un termine, assume come valore di ogni feature esattamente la *parola* legata al termine in questione, mediante il tipo di relazione specificata dalla feature. Nel caso in cui non sia presente alcuna relazione di questo tipo, viene definito un valore convenzionale NULL. Conviene illustrare il formalismo su dati reali, continuando sempre con l'esempio di partenza.

NORel	NP_PP

nohead	NULL
NULL	concorso
NULL	concorso
NULL	privilegi

Tra le righe riportate, quella al di sopra della linea costituisce il prototipo del feature vector, mentre quelle al di sotto costituiscono i dati veri e propri. La prima riga corrisponde a *concorso*, la seconda a *privilegi* e le ultime due entrambe a *crediti*, dal momento che tale testa è coinvolta in due differenti relazioni, entrambe con plausibilità 0.5. Si noti che con questo metodo di rappresentazione, il peso dato a *crediti* è doppio rispetto a quello dato a *privilegi*, in quanto al primo compete una sola riga mentre al secondo ne

competono due. Una tale situazione costituirebbe un fattore eccessivamente fuorviante per l'algoritmo di apprendimento: nell'algoritmo di conversione di formato (da XDG a L) si è così provveduto ad inserire una funzione di bilanciamento, che restituisce alle occorrenze dei candidati termini il loro effettivo valore, dedotto dal coefficiente di plausibilità

5.4 Modello globale-lessicale (G)

L'informazione rappresentata in questo modello è qualitativamente la medesima del precedente, differenziandosi però nella sua organizzazione. A differenza del modello L, si opera ora in un contesto globale. Tutte le occorrenze di una data testa in ingresso, ad esempio *crediti*, vengono raggruppate in una sola linea nel file di uscita. Questo equivale a fornire all'algoritmo di apprendimento una descrizione del comportamento *generale* di ciascun candidato termine, la quale viene ottenuta come somma di tutte le informazioni raccolte nel corpus di quel termine specifico. Il prototipo del feature vector è dato invece dalla sequenza di tutte le possibili combinazioni (o meglio, concatenazioni) relazione-parola individuate nel corpus. I vettori dati, infine, sono numerici e corrispondono alla somma delle plausibilità di volta in volta riscontrate in ciascuna occorrenza del termine, tenendo ovviamente distinti valori diversi per feature diverse. Riportiamo di seguito il relativo formato di uscita.

NORel-nohead	NP_PP-concorso	NP_PP-privilegi
1	0	0
0	1	0
0	0.5	0.5

In questo caso la prima riga corrisponde a *concorso*, la seconda a *privilegi* e la terza a *crediti*. Si noti che con il metodo fin qui descritto, termini con un numero maggiore di occorrenze tenderanno ad avere valori di feature (in generale) più alti. Si tratta di un comportamento non desiderabile, in quanto indice indiretto della frequenza con cui un termine compare: una tale informazione vuole essere intenzionalmente omessa dal modello. Per ovviare ad una tale “anomalia”, una volta completata la scansione del corpus, tutti i vettori vengono opportunamente normalizzati portando i loro moduli a lunghezza unitaria.

5.5 Modello globale-sintattico (S)

Questo modello è caratterizzato dal fatto di prendere in considerazione *esclusivamente* l’informazione *sintattica* associata alle relazioni che interessano i termini. Si distingue, infatti, dal precedente solamente per il tipo delle feature. Queste sono le stesse del modello (L), cioè rappresentano tutti i tipi di relazione possibili *indipendentemente* dall’elemento lessicale interessato dalla relazione.

Come per il modello precedente, il contesto preso in considerazione è globale: continuano quindi a valere le considerazioni circa la normalizzazione dei vettori. Il formato dei dati generati è il seguente:

NORel	NP-PP
1	0
0	1
0	1

Si noti che mentre la seconda riga è generata da una singola relazione con indice di plausibilità pari ad 1, la terza proviene dalla somma di due distinte relazioni, entrambe di plausibilità 0.5. Tale comportamento tende ad assimilare i termini poco ricorrenti, ma con plausibilità alta, a quelli molto ricorrenti, ma con indici di plausibilità minori. Questa circostanza porta ad includere nel modello una certa quantità di rumore, che potrebbe eventualmente essere filtrata escludendo tutti i campioni con plausibilità inferiore ad una soglia fissata.

5.6 Modello in frequenza (F)

Le motivazioni che hanno portato all'elaborazione di un modello basato sulla frequenza sono del tutto indipendenti dall'indagine specifica circa l'informazione contestuale. Intendiamo, in questo contesto, con frequenza il numero di occorrenze totali di ciascun termine nell'ambito del corpus: è evidente come tale informazione sia di natura radicalmente diversa da quella utilizzata nei precedenti modelli. Si consideri inoltre che le tecniche di normalizzazione dei modelli G ed S e la tecnica di ribilanciamento utilizzata nel modello L sono finalizzate, in ultima analisi, esattamente all'eliminazione di ogni dipendenza indiretta dalla frequenza dei rispettivi modelli. Le motivazioni di fondo che hanno portato alla definizione del modello F sono, quindi, di carattere prettamente comparativo. Come spesso riportato nella letteratura riguardante la TE, le prestazioni ottenute da processi computazionali basati su tecniche statistiche sono sorprendentemente buone. Si è pertanto ritenuto di procedere ad un confronto sperimentale dell'approccio seguito in questo studio, rappresentato dai modelli L, G ed S, con il modello che si presenterà in seguito.

Il feature vector del modello F è composto dall'unica componente utile, quella indicante il numero di occorrenze nel corpus del termine. Il formato ottenuto è quindi del tipo

```
Freq
----
1
1
1
```

Si noti che la frequenza relativa a *crediti* è 1 e non 2, come si potrebbe erroneamente pensare, in quanto le due apparizioni di *crediti* nel file di ingresso sono in realtà relative ad una medesima occorrenza, fatto denotato dalla presenza di un indice di plausibilità minore di uno. A tale proposito si osservi come il file di ingresso sia strutturato in blocchi caratterizzati localmente dalla medesima testa. L'unico modo corretto di calcolare il reale numero di occorrenze è quello di individuare opportuni sottoblocchi nei quali la somma delle plausibilità sulle singole righe sia *unitaria*.

5.7 modelli *ibridi*

Le evidenze sperimentali ottenute, come sarà mostrato nella presentazione dei risultati, hanno suggerito a posteriori l'elaborazione di tre ulteriori modelli di rappresentazione. L'informazione contenuta in tali modelli è qualitativamente la stessa di quella finora utilizzata. Essi sono costruiti sostanzialmente combinando i precedenti modelli, come suggerito dalla denominazione scelta: S+F, G+F e S+G+F. Dal punto di vista del formato dei feature vector, questi sono formati per esatta giustapposizione dei loro costituenti di base, e lo stesso vale evidentemente anche per i dati. Contrariamente ai quattro

modelli di base, non è stata prodotta alcuna routine automatica di generazione a partire dai dati grezzi, ma una generica funzione di “cucitura” che opera a partire da dati precedentemente elaborati secondo i modelli di base. Date le considerazioni fin qui svolte, il formato di uscita di questi tre modelli può essere agevolmente omesso.

Capitolo 6

Piattaforma sperimentale realizzata

6.1 Caratteristiche del corpus

Il corpus di riferimento scelto per le applicazioni sperimentali è costituito dal Codice Civile italiano. Trattandosi di un testo normativo-legale fondamentale, si suppone in esso non soltanto la definizione di un'ampia terminologia di base, ma anche il suo corretto e frequente utilizzo. Altro elemento essenziale, alla base di questa scelta, è stata la disponibilità di una terminologia controllata (*l'oracolo*) associata al corpus stesso.

L'importanza di tale fattore è stata precedentemente discussa in relazione al modello esteso del dominio e va, inoltre, notato come la disponibilità di tale risorsa costituisca un fatto di notevole importanza, data la sua rarità in altri domini. Operando tale scelta, si è anche voluto collocare questo studio in un quadro sperimentale normalizzato ed agevolmente riproducibile, nell'eventualità di ulteriori approfondimenti o di un successivo confronto con esperimenti affini.

Partendo dal corpus, sono state utilizzate due tecniche di parsing, distinte tra loro per l'uso del modulo di sottocategorizzazione verbale in CHAOS.

Con tale modalità sono stati quindi generati due diversi insiemi di apprendimento, denominati “*Lifuv*” e “*Zerolex*” (il primo dei quali con sottocategorizzazione). Ciascuno dei due insiemi è stato a sua volta suddiviso, mediante l’oracolo, in esempi positivi e negativi, per un totale di quattro file distinti. Riassumiamo in forma tabellare i valori quantitativi relativi al complesso dei dati disponibili.

Tabella 6.1: Insieme Lifuv, esempi positivi

Termini presenti	634
Occorrenze totali	35017
Frequenza minima	1
Frequenza massima	980
Frequenza media	55.2

Tabella 6.2: Insieme Lifuv, esempi negativi

Termini presenti	2493
Occorrenze totali	22899
Frequenza minima	1
Frequenza massima	2019
Frequenza media	9.2

Tabella 6.3: Insieme Zerolex, esempi positivi

Termini presenti	634
Occorrenze totali	34886
Frequenza minima	1
Frequenza massima	974
Frequenza media	55.0

Come è facilmente riscontrabile, in prima battuta i due insiemi di apprendimento appaiono sostanzialmente omogenei. Una grossa differenza va

Tabella 6.4: Insieme Zerolex, esempi negativi

Termini presenti	2475
Occorrenze totali	22679
Frequenza minima	1
Frequenza massima	2020
Frequenza media	9.2

invece notata nella distribuzione delle frequenze degli esempi positivi rispetto ai negativi. Mentre per i positivi si ha un valore di 55, il corrispondente dei negativi è soltanto 9. Questa sensibile differenza suggerisce, sin d'ora, una notevole capacità discriminante da parte dei modelli basati su frequenza, per i quali possiamo, quindi, ragionevolmente ipotizzare un'elevata efficienza.

6.2 Metodologia sperimentale

Si descrivono ora i criteri sperimentali generali ai quali si è fatto riferimento al fine di generare risultati realmente attendibili. Nella parte generale, riguardante l'apprendimento di alberi di decisione, si è già diffusamente discusso della particolare attenzione necessaria per la generazione di insiemi di addestramento e di test significativi. Un punto di partenza, decisamente irrinunciabile per qualsiasi prova sperimentale, è certamente il mescolamento casuale dell'insieme generale di apprendimento, prima che questo venga suddiviso in test set e training set.

6.2.1 n -fold cross validation

Nonostante un grado di confidenza intuitivamente accettabile, anche il mescolamento casuale potrebbe nascondere disomogeneità o distribuzioni anomale degli oggetti. Si è dunque deciso di fare riferimento ad una tecnica mag-

giormente affidabile nota come *n-fold cross validation*. Tale tecnica consiste, essenzialmente nella suddivisione dell'insieme generale di apprendimento (già mescolato) in n distinti folds contenenti indifferentemente esempi positivi e negativi. Vengono eseguite, quindi, n prove distinte del *medesimo* esperimento, in ciascuna delle quali il ruolo di test set è ricoperto *a turno* dagli n fold. In ogni prova, il ruolo di insieme di apprendimento è ricoperto invece dall'unione degli $n - 1$ folds non utilizzati per il test. Al termine dell'esecuzione prove, si saranno ottenuti n distinti risultati. La media di questi costituisce un valore sufficientemente affidabile, che viene presentato come risultato complessivo dell'esperimento.

6.2.2 Variazione progressiva degli esempi negativi

Fermo restando l'uso della tecnica appena descritta, si è ravvisata la necessità di una indagine ulteriore che ha lievemente complicato la procedura da utilizzare. Nella spiegazione relativa agli obiettivi generali di studio, si è espressamente affrontato il problema degli esempi negativi. Se dal punto di vista dell'apprendimento la loro presenza è assolutamente necessaria, come è stato sottolineato nella presentazione degli alberi di decisione, va anche considerato l'aspetto puramente linguistico della questione. La disponibilità di evidenze negative è una risorsa intrinsecamente rara e, da un certo punto di vista, anche "innaturale". È stato perciò stabilito di introdurre nella *n-fold cross validation* un ulteriore grado di libertà, costituito dalla variazione del rapporto tra esempi positivi e negativi.

Nello specifico, viene mantenuto costante il numero di esempi positivi nel training set, mentre quello dei negativi varia da un valore minimo (stabilito dalla dimensione del fold) a quello massimo consentito. In questo modo il numero di iterazioni necessarie cambia n a $n(n - 1)$ implicando un notevole

sfuerzo computazionale, ma ottenendo un'elevata qualità dei risultati, come verrà illustrato in seguito. Passiamo dunque a descrivere formalmente il procedimento generale di sperimentazione.

6.2.3 Formalizzazione della procedura generale

Fissato un determinato modello ed opportuni vincoli di plausibilità e di frequenza, il generatore di modelli produce due files distinti contenenti rispettivamente l'insieme P degli esempi positivi e l'insieme N degli esempi negativi. La metodologia adottata mira a verificare l'andamento dell'efficienza di apprendimento al variare della proporzione tra esempi positivi e negativi. Una volta fissato un n opportuno, si effettua un partizionamento dei due insiemi, tale che

$$\begin{aligned}P &= P_1 \cup P_2 \cup \dots \cup P_i \cup \dots \cup P_n \\N &= N_1 \cup N_2 \cup \dots \cup N_i \cup \dots \cup N_n\end{aligned}$$

Si eseguono quindi $n - 1$ distinte prove sperimentali, ciascuna delle quali viene iterata n volte al fine di calcolarne la media. Tra una prova e l'altra viene quindi fatta variare la quantità di esempi negativi sottoposta al sistema di apprendimento. La quantità di esempi positivi, viceversa, è mantenuta costante.

Per ogni prova $j \in \{1, \dots, n - 1\}$ e per ogni iterazione $i \in \{1, \dots, n\}$ restano così definiti l'insieme di test $T_{i,j}$ e l'insieme di addestramento (*training set*) $A_{i,j}$:

$$T_{i,j} = P_i \cup N_i$$

$$A_{i,j} = \bigcup_{r \neq i} P_r \cup \bigcup_s N_s \quad s = i + 1, \dots, i + j \text{ mod } n$$

Il risultato della singola iterazione i nella prova j sarà il valore percentuale di efficienza $e_{i,j}$ ottenuto nella valutazione di $T_{i,j}$. Il risultato complessivo della j -esima prova sarà dato, infine, dalla media aritmetica delle sue n iterazioni:

$$E_j = \frac{1}{n} \sum_{i=1}^n e_{i,j}$$

6.3 Software realizzato

Presentiamo in questa sezione i vari moduli software realizzati per l'elaborazione dei modelli e delle tecniche sperimentali già ampiamente discussi. Tra le molte opzioni possibili, è stato scelto l'utilizzo del linguaggio PERL per le sue caratteristiche peculiari in relazione alla manipolazione dei file di testo e delle stringhe in generale. Dal punto di vista puramente operativo, il compito di generare modelli o predisporre training set distinti si configura, infatti, come una pura operazione di riscrittura. Nel primo caso verrà variato il contenuto dei files nella forma, mentre nel secondo si opereranno dei tagli dal punto di vista della quantità di esempi generati in uscita. Le descrizioni che seguiranno intendono dare un quadro generale del software prodotto per l'aspetto funzionale. I dettagli algoritmici sono reperibili direttamente all'interno del codice commentato, che è interamente riportato in appendice.

6.3.1 Generatore di modelli “*filtro.pl*”

È il principale e il più articolato tra i moduli software realizzati. Come il nome suggerisce, si tratta di un filtro avente il compito di tradurre il formalismo XDG in una rappresentazione riconoscibile dal sistema C4.5, in termini di descrizione di oggetti come feature vectors. Caratteristica essenziale del programma sono i suoi parametri di configurazione. È possibile selezionare uno tra i modelli L, G, S, F come formato dei dati in uscita ed è altresì possibile imporre dei tagli ai dati in ingresso relativamente alla soglia di plausibilità e alla minima e massima frequenza accettata. I dati di ingresso vengono letti separatamente dai due files di esempi positivi e negativi. I files prodotti rispondono ai requisiti formali richiesti da C4.5 e presentano dunque estensioni .names (descrizione del formato dei dati) e .data (dati effettivi). I dati prodotti da *filtro.pl* sono immediatamente elaborabili da C4.5 ma, come si è detto nei criteri generali di sperimentazione, essi verranno suddivisi in più computazioni successive al fine di ottenere risultati maggiormente attendibili. Si noti in particolare che vengono di fatto generati due distinti files .data, contenenti rispettivamente gli esempi positivi e negativi, al fine di realizzare gli esperimenti con l'aumento progressivo degli esempi negativi. Nel modulo *filtro.pl* è stata inoltre inclusa un'utile funzione di analisi generale dei files in ingresso. Tale funzione è stata utilizzata per il calcolo delle tabelle con la descrizione del corpus presentate in precedenza.

6.3.2 Iteratore computazionale “*iter.pl*”

Questo modulo software si occupa specificamente della generazione iterativa dei learning set e dei test set da sottoporre a C4.5 partendo da un medesimo insieme di dati. La procedura seguita è, a seconda dei casi, la n -fold

cross validation *standard* oppure quella più generale con variazione graduale della quantità di esempi negativi. Il modulo riceve in ingresso i due files `.data` e un parametro corrispondente al numero di fold desiderati. Un terzo parametro opzionale permette la selezione della modalità operativa. Altro importante compito di questo modulo è quello di interfacciarsi direttamente con C4.5, gestendone le invocazioni, il passaggio dei dati e la raccolta dei risultati ottenuti per ciascuna iterazione. Man mano che questi si rendono disponibili, *iter.pl* si occupa della generazione dei risultati finali procedendo al calcolo dei valori medi. Il modulo si occupa infine della generazione di un report complessivo dell'esperimento comandato, che viene memorizzato su un apposito file.

6.3.3 Script di gestione dei lavori “*batch*”

Si tratta semplicemente di uno shell script UNIX, modificabile a piacimento, avente la funzione di serializzare l'esecuzione di più esperimenti diversi programmati, variando i parametri. Si pone quindi al livello più alto della piattaforma sperimentale proposta e si interfaccia direttamente con il modulo *iter.pl*, con il quale concorre alla generazione del report finale.

Capitolo 7

Risultati sperimentali

Nel precedente capitolo ci siamo occupati di definire un quadro formale di riferimento nel quale inserire l'esecuzione delle prove sperimentali. Passiamo ora alla descrizione particolareggiata degli esperimenti condotti e alla presentazione dei risultati ottenuti. Il criterio di presentazione seguito è volto ad illustrare non soltanto i dati numerici di volta in volta ottenuti ma anche il percorso evolutivo che ha portato, gradualmente, all'abbandono di determinati modelli (il locale, in particolare) e allo sviluppo di nuovi (gli "ibridi") nella misura in cui questi si rendevano via via più promettenti in relazione alla nostra indagine sull'informazione contestuale.

7.1 Valutazioni preliminari

Presentiamo qui i risultati di un primo esperimento condotto in una fase preliminare della valutazione dei modelli. Sono stati valutati entrambi gli insiemi di riferimento con un taglio di massima ambiguità uguale a 4, nessun taglio sulla frequenza, numero di folds $N=10$.

Avremo modo di commentare ampiamente il confronto tra le tre rappresentazioni su dati maggiormente particolareggiati. Si vuole qui discutere esclusivamente la prestazione particolarmente scadente del modello L. La

Tabella 7.1: Tassi di errore riscontrati per i tre modelli di base

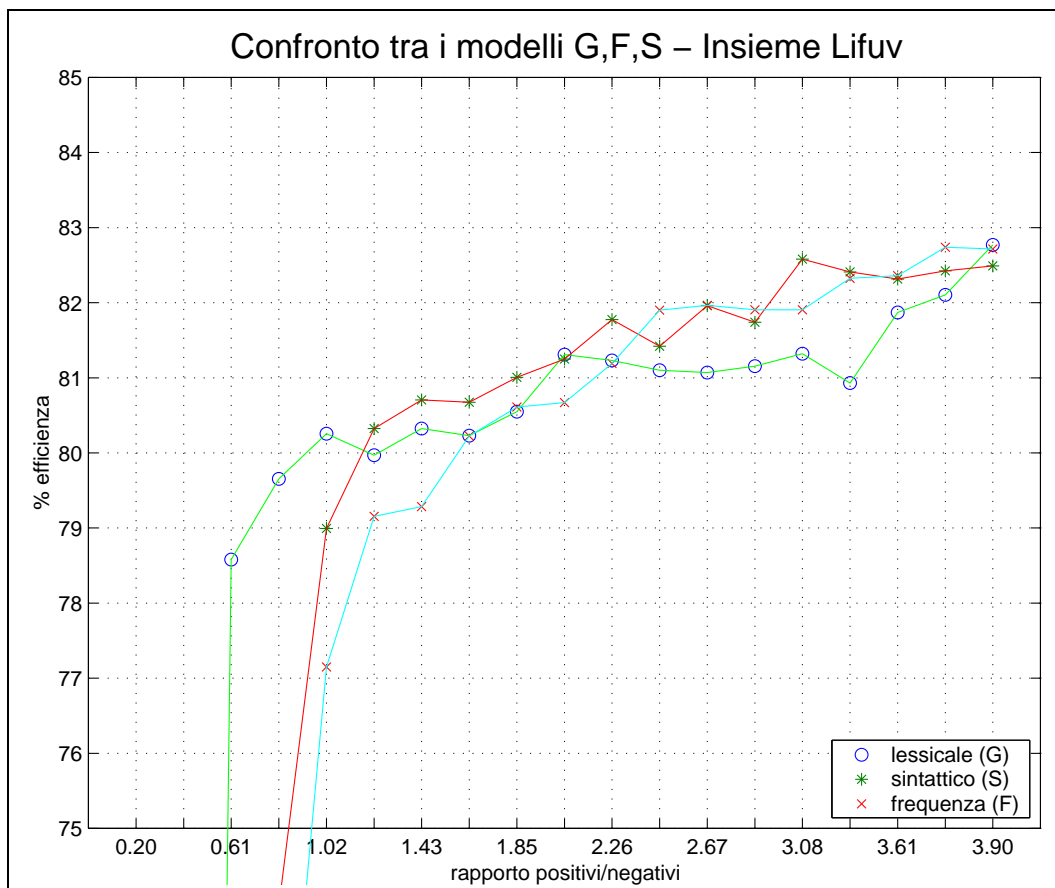
Termini/Modello	L	G	S
Lifuv	27.93%	18.42%	17.43%
Zerolex	28.73%	20.5%	17.7%

ragione dell'elevato tasso di errore risiede, con ogni probabilità, in un'inadeguatezza strutturale del modello stesso. Questo tenta, infatti, di dare una grande quantità di differenti descrizioni di termini basandosi su informazioni molto scadenti, in quanto relative a *singole* apparizioni dei termini nel corpus. Ricordiamo che nel modello L non è prevista alcuna funzione di raggruppamento. Date le precedenti valutazioni, da questo momento in poi, il modello L verrà escluso da ulteriori esperimenti. Un'ultima osservazione sui dati riportati: l'insieme Lifuv, frutto di una elaborazione più raffinata dal punto di vista linguistico, presenta tassi di errore costantemente inferiori allo Zerolex.

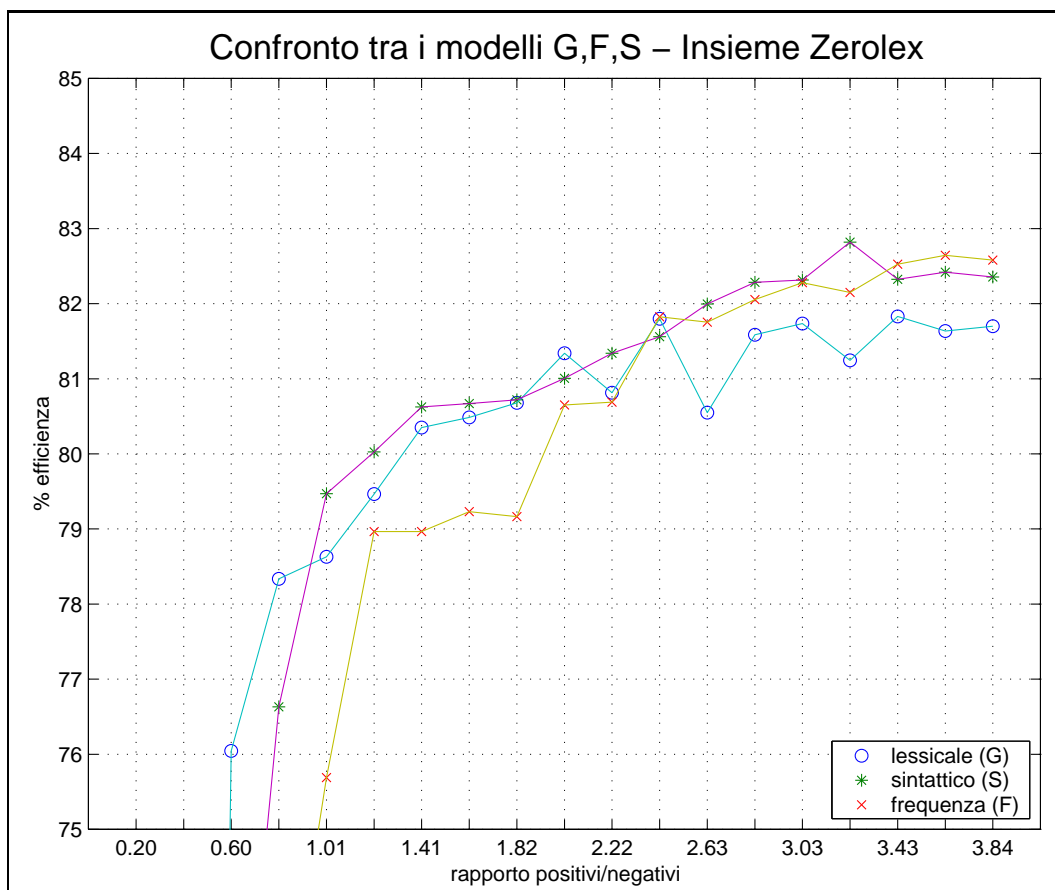
7.2 Variazione progressiva esempi negativi

Presentiamo ora i risultati derivanti dalla tipologia di esperimenti più articolata, vale a dire quella basata sulla combinazione tra n -fold cross validation e variazione graduale degli esempi negativi. Introduciamo anche una prima valutazione del modello F, che viene confrontato con S e G. Gli esperimenti sono stati condotti su entrambi gli insiemi terminologici, utilizzando un vincolo di massima ambiguità pari a 5 e nessuna restrizione sulla frequenza. Il numero di folds utilizzato è pari a 20 valore, questo, decisamente "impegnativo" sul piano computazionale, che permette comunque una piena comprensione dell'andamento generale del fenomeno. I risultati sono presen-

tati direttamente in forma grafica e rappresentano l'andamento dell'efficienza generale dell'apprendimento al variare della proporzione tra esempi negativi e positivi.



Dall'esame dei grafici riportati emergono alcune importanti osservazioni. Innanzitutto, tutte le curve presentano un marcato cambio di tendenza in prossimità del punto in cui le quantità di esempi positivi e negativi si equivalgono, e cioè per valori delle ascisse prossimi ad 1. Tale comportamento indica da un lato l'assoluta necessità di includere esempi negativi nella fase di apprendimento come era già noto, e dall'altro la loro progressiva diminuzione di importanza al superamento di una determinata soglia. La tendenza



delle curve ad appiattirsi indica chiaramente una stabilizzazione del processo. Confrontando i due insiemi di termini, il Lifuv presenta una maggiore rapidità di salita in corrispondenza di un numero di esempi negativi ancora esiguo, fatto che conferma senza dubbio la sua superiorità qualitativa rispetto allo Zerolex.

Dal punto di vista dei modelli rappresentati, si osserva una prestazione abbastanza deludente del modello G quello di gran lunga più complesso, contenente informazioni di tipo lessicale. È però da apprezzare la sua notevole rapidità nel migliorare l'efficienza, particolarmente evidente nell'insieme Lifuv. Veniamo ora all'aspetto più interessante: il confronto tra i modelli S

ed F. S offre in generale prestazioni migliori rispetto alla frequenza per gran parte del grafico tranne che nella parte finale, in cui F chiude costantemente con risultati migliori. Nell'analisi del corpus a nostra disposizione avevamo già ipotizzato un grosso potere discriminante da parte della frequenza a causa dell'elevata differenza in media di occorrenze tra gli esempi positivi e negativi (55 contro 9), e i dati ottenuti confermano nettamente le attese.

È dunque estremamente importante l'essere riusciti a riprodurre, ed in parte superare, la prestazione di F con un modello (S) basato esclusivamente su informazione contestuale, nel caso specifico quella puramente sintattica. *Questo fatto costituisce certamente il risultato più rilevante ottenuto nel presente studio ed afferma, senza alcun dubbio, il valore qualitativo dell'informazione contestuale nel riconoscimento dei termini.*

7.3 Confronto con i modelli ibridi

I risultati ottenuti nel precedente esperimento hanno stimolato un ulteriore approfondimento del confronto tra il modello F e quelli basati su informazione contestuale. Come descritto nella parte modellistica, sono stati creati ad-hoc 3 nuovi modelli tramite semplice composizione dei precedenti. I nuovi modelli sono dunque S+F, G+F e S+G+F. Obiettivo degli esperimenti relativi è ora quello di valutare l'eventuale incremento di efficienza di F nel momento in cui gli viene fornito *in aggiunta* un contenuto informativo di tipo contestuale.

Alcuni risultati preliminari sono raccolti in forma tabellare, relativamente ad un esperimento con $n=10$, nessun vincolo di frequenza, massima ambiguità pari a 5. L'insieme utilizzato è il Lifuv e il confronto è condotto contemporaneamente sui modelli di base e sugli ibridi.

I dati raccolti nella tabella presentano una chiara tendenza nel passaggio

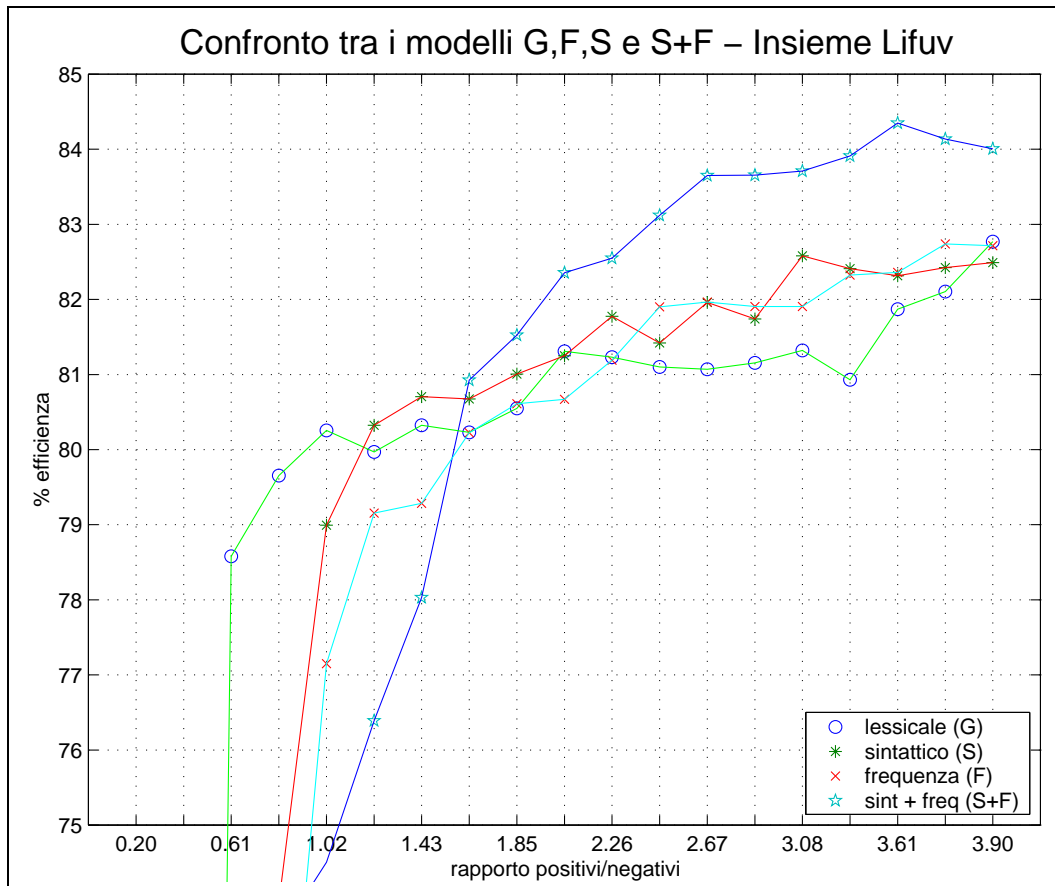
Tabella 7.2: Tassi di errore nel confronto tra modelli di base e ibridi

S+F	G+F	S+G+F	G	S	F
16.43%	18.29%	17.59%	16.91%	17.17%	17.33%

dai modelli semplici a quelli ibridi: l'aggiunta di informazione contestuale al modello F risulta efficace nel solo caso puramente sintattico (S), mentre il modello lessicale (G) continua a presentare comportamenti al di sotto delle aspettative. D'altra parte il miglioramento ottenuto da F, nel passaggio ad S+F, è un dato estremamente rilevante e conferma il successo ottenuto dal modello S assoluto, già evidenziato nei grafici precedenti. La rappresentatività di questo esperimento è abbastanza ridotta in quanto limitata ad un solo punto di calcolo, quello relativo cioè alla massima partecipazione di esempi negativi all'apprendimento.

Si è dunque proceduto ad un ulteriore confronto dei modelli S ed S+F. Le condizioni sperimentali sono le medesime utilizzate nei precedenti casi (già illustrati graficamente). Si consideri quindi il grafico riportato nel seguito.

Il contenuto di quest'ultimo grafico costituisce una netta e ponderata conferma della superiorità dell'informazione S rispetto a quella F. Si noti infatti che il modello S+F supera le prestazioni finora ottenute in qualsiasi altro contesto sperimentale. In particolare, il superamento del modello F *assoluto* conferisce valore pieno alla precedente affermazione.



Capitolo 8

Conclusioni

Nel corso del presente studio è stata innanzi tutto formulata una precisa ipotesi sul piano teorico: l'applicabilità di uno specifico strumento dell'apprendimento automatico (gli alberi di decisione) ad una ben definita applicazione di natural language processing, cioè l'estrazione di terminologia a partire da un corpus di testi.

La prima parte della trattazione si è quindi preoccupata di stabilire l'effettiva plausibilità di tale ipotesi sul piano prettamente teorico, approfondendo i concetti generali di entrambe le discipline ad un livello almeno sufficiente a coglierne i potenziali aspetti problematici dal punto di vista modellistico.

La successiva fase sperimentale ha dato piena conferma della necessità di un approccio applicativo fondato su salde basi teoriche, infatti la conferma empirica delle ipotesi di partenza è giunta solo dopo sforzi parzialmente inattesi e dopo la definizione di ulteriori modelli inizialmente non previsti.

Scopo specifico degli esperimenti era la ricerca di una evidenza quantitativa dell'efficacia dell'informazione testuale riguardante il contesto sintattico dei termini. Alcune peculiarità dell'ambiente sperimentale sono tuttavia risultate essere particolarmente favorevoli a tecniche, anch'esse studiate quan-

titativamente, basate su un approccio statistico anziché prettamente linguistico. Tale situazione di apparente stallo è stata superata per via indiretta, cioè con l'elaborazione di uno specifico modello in cui l'informazione sintattica si affiancasse a quella statistica. La superiore efficacia applicativa di tale modello *ibrido* rispetto a quello puramente statistico ha in conclusione dato piena conferma delle ipotesi iniziali.

Appendice A

Codice sorgente commentato

A.1 Codice di *filtro.pl*

```
#!/usr/bin/perl

# Parametro di tolleranza per ovviare all'approssimazione dei valori di
# plausibilità nei file di input.

$epsilon = 0.01;

# numexTot conta gli esempi totali forniti. Parte da -1 in accordo con
# l'aggiunta di una riga di terminazione ad ./examples.sorted

$numexTot = -1;

# Calcolo del minimo comune multiplo tra due numeri

sub mcmfunc($$) {

    my ($num1, $num2) = @_;

    if ($num1 > $num2) {
```

```

$temp = $num1;
    $temp2 = $num2;
}
else {
    $temp = $num2;
    $temp2 = $num1;
}
$numero = $temp;
while ($numero/$temp2 - int($numero/$temp2) != 0) {
$numero += $temp;
}
return $numero;
}

# Fusione dei due file di esempi positivi e negativi. Su ciascuna riga viene
# aggiunto inoltre il tag di classificazione ("Pos" o "Neg").
# Genera il file ./examples

sub MergeExamples($$) {

    my ($infile1, $infile2) = @_;

    open (OUTPUT, ">examples");
    open (GINPUT, "<$infile1") || die "Can't input $infile1 $!";

    # Lettura del file degli esempi positivi
    while (<GINPUT>) {

@linea = split;

        # Ciascuna occorrenza di ',' viene fatta precedere dal carattere '\'.
        $linea[2] =~ s/,/\\",/;
        $linea[4] = "Pos\n";
        # Aggiunge un carattere di tabulazione a fine riga e scrive su file.
        $str = join("\t",@linea);
        print OUTPUT $str;
    }
    close GINPUT;

    open (BINPUT, "<$infile2") ||
        die "Can't input $infile2 $!";

```

```
# Lettura del file degli esempi negativi. Analogo al precedente.
while (<BINPUT>) {

    @linea = split;
    $linea[2] =~ s/,/\\",/;
$linea[4] = "Neg\n";
    $str = join("\t",@linea);
#Stampa sul file di output
    print OUTPUT $str;
}

# Chiusura dei file aperti

    close BINPUT;
    close OUTPUT;
}

# Sort del file ./examples basato esclusivamente sul campo header (il primo).
# Le righe del file sono quindi ordinate "per blocchi", lasciando inalterata
# la struttura locale di ciascun esempio. Molti algoritmi utilizzati assumono
# questa struttura del file. Genera ./examples.sorted e aggiunge una riga di
# terminazione.

sub sortdata($) {
    my ($infile) = @_ ;
    my @vect;
    my @svect;

    if (! -r $infile) { die "Can't read input $infile\n";}
    if (! -f $infile) { die "Input $infile is not a plain file\n";}
    open (SINPUT, "<$infile") || die "Can't input $infile $!";

    $index = 0;
    $plaustot = 0;
    $position = 0;

    # Legge file di input, crea un vettore con offset delle posizioni

    while (<SINPUT>) {
@linea = split;
```

```
#Si tiene conto della plausibilita' parziale, per fare in modo che gli
#esempi appartenenti ad un contesto non si mescolino con gli altri
#ma restino in sequenza localmente.

$plautot += @linea[3];
$ambig++;
if ($plautot >= 1-$epsilon) {
    # Mantiene in un vettore le posizioni nel file di ogni nuovo
    # header, con il relativo coefficiente di ambiguità.
    $vect[$index] = @linea[0]."\t".$position."\t".$ambig;
    $ambig = 0;
    $index++;
    $plautot = 0;
    # Memorizzamo la posizione dell'header nel file
    $position = tell(SINPUT);
}
}

$num = $index;

# Ordinamento del vettore delle posizioni
@svect = sort @vect;

    open(SOUTPUT, ">$infile.sorted");
    for ($index = 0; $index <= $num; $index++) {
$_ = $svect[$index];
@riga = split;

        seek SINPUT,@riga[1],0;

# Scrittura sul file di output degli esempi ordinati.

for ($j = 0; $j < @riga[2]; $j++) {
    $temp = <SINPUT>;
    print SOUTPUT $temp;
}
}

# Aggiunge al file una riga di terminazione. E' in accordo con il valore
# iniziale numexTot=-1
```

```
print SOUTPUT "flush\tNORel\tandstop\t1.0\tNeg\n";

close SINPUT;
close SOUTPUT;
}

# Raccoglie una distribuzione generale delle frequenze nel file di ingresso,
# tenendo separati esempi positivi e negativi.

sub FreqDistr($) {

    my ($infile) = @_ ;
    my $example;
    my $extot = 0;
    my $card = 0;

    if (! -r $infile) { die "Can't read input $infile\n";}
    if (! -f $infile) { die "Input $infile is not a plain file\n";}
    open (FINPUT, "<$infile") || die "Can't input $infile $!";

    while (<FINPUT>) {
        @linea = split;

        $exflag += @linea[3];
        if ($exflag >= 1-$epsilon) {
            $exflag = 0;
            if (@linea[1] ne "NORel") {
                $extot++;
                $example = @linea[0]."-".@linea[4];
                if (! exists $fdhash{$example}) {
                    $card++;
                    $fdhash{$example} = $card;
                }
                $fdvect[$fdhash{$example}]++;
            }
        }
    }

    #Togliere i commenti per lavorare con i valori permille
    #for ($i = 1; $i <= $card; $i++) {
```

```

#$fdvect[$i] = $fdvect[$i]*1000/$extot;
#}
close FINPUT;
}

# Valuta se un esempio è accettabile secondo i tagli di frequenza impostati

sub FreqOk($$) {

    my $example = @_[0]."-".@[1];

    #print $example.": ".$fdvect[$fdhash{$example}];
    if ( ($fmin eq "min") || ($fdvect[$fdhash{$example}] >= $fmin) ) {
if ( ($fmax eq "max") || ($fdvect[$fdhash{$example}] <= $fmax) ) {
        #print ": accettata\n";
        return 1;
    }
    }
    #print": respinta\n";
    return 0;
}

# Stampa modalità di utilizzo ed esce

sub helpuser {

    print "\n Uso: $0 Modalità File-pos File-neg MaxAmbig Fmin Fmax NomeFile [-q]\n";
    print "  Modalità:\n\t\t-l: locale-lessicale\n\t\t-g: globale-";
    print "lessicale\n\t\t-s: sintattica\n\t\t-f: frequenza\n\t\t";
    print "-a: analisi di frequenza (richiede solo un input file)\n\n";
    print "  File-pos: file di esempi positivi\n\n";
    print "  File-neg: file di esempi negativi\n\n";
    print "  MaxAmbig: valore di massima ambiguità\n\n";
    print "  Fmin      : frequenza inferiore di taglio\n\n";
    print "  Fmax      : frequenza superiore di taglio\n\n";
    print "  NomeFile: radice nome files in uscita (.names, .data)\n\n";
    die "          -q: output ridotto\n\n";
}

```

```
# Filtra il canale output su video

sub msgout($) {

    my ($string) = @_;

    if ($modex ne "-q") {
print $string;
    }
}

# Programma principale. Genera i files in formato .names e .data a partire
# dagli esempi positivi e negativi, secondo le modalità impostate su linea
# di comando.

if ( ($#ARGV < 4) && (@ARGV[0] ne "-a") ) {
    helpuser;
}

($a1, $a2, $a3, $a4, $a5, $a6, $a7, $a8) = @ARGV;

if ( ($a1 eq "-l") || ($a1 eq "-g") || ($a1 eq "-s") || ($a1 eq "-f") ) {

    $context = $a1;
    $gfile = $a2;
    $bfile = $a3;
    $maxambig = $a4;
    $fmin = $a5;
    $fmax = $a6;
    $outf = $a7;
    $modex = $a8;

}

elsif ($a1 eq "-a") {

    $context = $a1;
    $gfile = $a2;
    $maxambig = $a3;
```

```
$fmin = $a4;
$fmax = $a5;

}

else { helpuser; }

# Impostazione di un valore di tolleranza per ovviare all'approssimazione dei
# valori di plausibilità nei files di ingresso

$soglia = 1/($maxambig) - 0.00001;

if ($context ne "-a") {

    msgout "\nFLT : Fusione degli esempi positivi e negativi...\n";
    MergeExamples($gfile,$bfile);
    msgout "FLT : Fusione eseguita.\n";
    msgout "\nFLT : Ordinamento del file esempi...\n";
    sortdata("./examples");
    msgout "FLT : Ordinamento eseguito.\n";
    msgout "FLT : Analisi delle frequenze...\n";
    FreqDistr ("./examples.sorted");
    msgout "FLT : Analisi eseguita.\n";

    $mfile="./examples.sorted";
    if (! -r $mfile) { die "Can't read input $mfile\n";}
    if (! -f $mfile) { die "Input $mfile is not a plain file\n";}
    open (INPUT, "<$mfile") || die "Can't input $mfile $!";

    $mode=">";
    open (OUTPUT1, "$mode$outf.names") || die "Can't output $outf.names $!";
    open (OUTPUT2G, "$mode$outf"."g.data") || die"Can't output $outfg.data $!";
    open (OUTPUT2B, "$mode$outf"."b.data") || die"Can't output $outfb.data $!";
    open (RESULTS, ">>results");
}

# Esecuzione in modalita' ** GLOBALE-LESSICALE **

if ($context eq "-g") {

    print "FLT-G: Generazione esempi\n";
```

```
# bffhash: tabella hash per per tenere traccia delle features considerate

my %bffhash;
$count = 0;
$righebuone = 0;
$numex = 0;
$oldh = "NULL";
$oldc = "NULL";
$coin = 0;
$gtot = 0;
$btot = 0;

# Generazione del file .names secondo il formato richiesto da C4.5

msgout "\nFLT-G: Scrittura del file .names...\n";
print OUTPUT1 "Pos, Neg.\n\n";
while (<INPUT>) {
@linea = split;

# Determina il numero di esempi totali.
# Se ha trovato un nuovo esempio incrementa il counter generale

if ( (@linea[0] ne $oldh) ||
      ( (@linea[0] eq $oldh) && (@linea[4] ne $oldc) ) ) {
    $numexTot++;
    $oldh = @linea[0];
    $oldc = @linea[4];
    $coin = 1;
}

if ( (@linea[1] ne "NOrel") && (@linea[3] >= $soglia) &&
      (FreqOk(@linea[0], @linea[4])) ) {

    $righebuone++;

    # L'esempio ha almeno una riga valida. Incrementa counter.
    if ($coin == 1) {
$coin = 0;
$numex++;
    }
}
}
```

```
#Determinazione della feature,
$feature = @linea[1]."-".@linea[2];

# Se si tratta di una nuova feature allora viene inserita
# nella tabella hash insieme al suo indice progressivo

if(! exists $bfhash{$feature}) {
$bfhash{$feature} = $count++;
print OUTPUT1 $feature.": continuous.\n";
}
}

msgout "FLT-G: Esempi Validi/Totali: ".$numex."/".$numexTot."\n";
printf (RESULTS "Ex. Val/Tot: %5d/%5d\n",$numex, $numexTot);
msgout "FLT-G: Scrittura eseguita.\n";
msgout "\nFLT-G: Numero di features selezionate: ".$count."\n";

#Si posiziona all'inizio del file di Input.
seek INPUT,0,0;

# Si dichiara una nuova tabella hash per la memorizzazione degli header
my %bhash;

$count=0;

# Generazione del file .data (rappr. ** GLOBALE-LESSICALE **)

$oldh = "NULL";
$oldc = "NULL";
msgout "\nFLT-G: Scrittura del file .data...\n";
$numrighe = 0;
$coin = 0;

# Prepara un feature vector nullo
for ($j = 0; $j < $count; $j++) {
$bvect[$j] = 0;
}

while (<INPUT>) {
@linea = split;
```

```
# Considera il precedente esempio letto dal file. Se era stato
# considerato valido allora lo elabora e lo salva sul file in uscita

if ( ($coin == 1) && ($oldh ne "NULL") && (
    (@linea[0] ne $oldh) ||
    ( (@linea[0] eq $oldh) && (@linea[4] ne $oldc)) ) ) {

    $coin = 0;
    # Calcola la somma delle componenti del vettore esempio
    $somma=0;
    for ($j = 0; $j < $count; $j++) {
    $somma += $bvect[$j];
    }

    # Scrive su file il vettore esempio normalizzato
    for ($j = 0; $j < $count; $j++) {
if ($j != 0) {
    if ($bvect[$count] eq "Pos") {
print OUTPUT2G ", ";
    }
    else {
print OUTPUT2B ", ";
    }
}
if ($bvect[$count] eq "Pos") {
    print OUTPUT2G ($bvect[$j]/$somma);
}
else {
    print OUTPUT2B ($bvect[$j]/$somma);
}
}

    # Scrive la classificazione relativa (Pos/Neg)
    if ($bvect[$count] eq "Pos") {
print OUTPUT2G ", $bvect[$count].\n";
    $gtot++;
    }
    else {
print OUTPUT2B ", $bvect[$count].\n";
    $btot++;
}
}
```

```

    }
    # Pulisce il feature vector per un nuovo esempio
    for ($j = 0; $j < $count; $j++) {
$bvect[$j] = 0;
    }
}

# Considera la riga corrente. Se appartiene ad un esempio valido
# allora la prende in considerazione ed incrementa il valore
# cumulativo della feature corrispondente

if ( (@linea[1] ne "NORel") && (@linea[3] >= $soglia) &&
    (FreqOk(@linea[0], @linea[4])) ) {
    $numrighe++;
    $coin = 1;

    # Stampa percentuali di avanzamento
    if ($modex ne "-q") {
msgout int(($numrighe*100)/($righebuone))."% \r";
    }

    $feature= @linea[1]."-".@linea[2];

    # Calcola il numero di occorrenze della feature
    $bvect[$bfhash{$feature}] += @linea[3];
    $bvect[$count] = @linea[4];
}
$oldh = @linea[0];
    $oldc = @linea[4];
}
msgout "FLT-G: Scrittura eseguita.\n";
print "FLT-G: Positivi: $gtot Negativi: $btot\n";
}

# Esecuzione in modalita' ** LOCALE-LESSICALE **

elsif ($context eq "-l") {

    print "FLT-L: Generazione esempi\n";

    my %lfhash;

```

```

$count = 0;
$plaustot=0;
$ambig=0;
$mcm=1;
$numex = 0;
$exflag = 0;
$gtot = 0;
$btot = 0;

# Rappresentazione lessicalizzata: analisi esempi e scrittura del .names

print OUTPUT1 "Pos, Neg.\n\n";
msgout "\nFLT-L: Analisi degli esempi in input...\n";
while (<INPUT>) {
    @linea = split;

# Conta gli esempi presenti nel file
$exflag += @linea[3];
if ($exflag >= 1-$epsilon) {
    $numexTot++;
    $exflag = 0;
}

# Se sono valide relazione e soglia procede
    if ( (@linea[1] ne "NORel") && (@linea[3] >= $soglia) &&
        (FreqOk(@linea[0], @linea[4])) ) {
        # Tiene traccia della plausibilita' totale
        $plaustot += @linea[3];
        $ambig++;
        if ($plaustot >= 1-$epsilon) {
            $numex++;
        }
        # Salva il vecchio minimo comune multiplo
        $mcmold = $mcm;
        # Calcola il nuovo minimo comune multiplo
        $mcm = mcmfunc($mcm,$ambig);

#Togliere i commenti per controllare il calcolo del mcm.
#if ($mcmold != $mcm) {
#    print $mcmold." ".$ambig." ".$mcm."\n";
#}

```

```

$plaustot = 0;
$ambig = 0;
}

# Determina la feature corrente.

    $feature = @linea[1];
$cond = 1;

# Se è una nuova feature la inserisce nella tabella lfhash
# insieme al suo indice progressivo.

    if(! exists $lfhash{$feature}) {
$lfhash{$feature} = $count;
$lvect[$count++] = $feature.".";
$cond = 0;
}

# Concatena la feature con la parola in relazione con l'header

$feat = $feature."-".@linea[2];

# Se la coppia feature-parola non è stata già presa in
# considerazione la inserisce nella tabella lhhash

    if (! exists $lhhash{$feat}) {
$lhhash{$feat} = "y";
# Se l'elemento preso in considerazione e' il primo...
    if ($cond == 0) { $prestring=" "; }
# Se non e' il primo...
elseif ($cond == 1) { $prestring=", "; }
# Prepara il vettore per il .names
# Ottimizzabile? Ver. se si può scrivere ad ogni cambio head.
$lvect[$lfhash{$feature}] =
    $lvect[$lfhash{$feature}].$prestring.@linea[2];
}
}

}

msgout "FLT-L: Analisi eseguita.\n";
msgout "FLT-L: Esempi Validi/Totali: ".$numex."/".$numexTot."\n";
printf (RESULTS "Ex. Val/Tot: %5d/%5d\n",$numex, $numexTot);

```

```

msgout "FLT-L: Fattore moltiplicativo di ambiguità (mcm): ".$mcm."\n";
msgout "FLT-L: Numero di features selezionate: ".$count."\n";
msgout "\nFLT-L: Scrittura del file .names...\n";

# Scrittura file .names (visualizzazione percentuale progressiva).

for ($i = 0; $i < $count; $i++) {
if ($modex ne "-q") {
    msgout int(($i*100)/($count-1))."% \r";
}
}
print OUTPUT1 $lvect[$i].", NULL.\n";
}

msgout "FLT-L: Scrittura eseguita.\n";
seek INPUT,0,0;

# Generazione del file .data (rappr. ** LOCALE-LESSICALE **)

msgout "\nFLT-L: Scrittura del file .data...\n";
$wrtlines = 0;
# Lettura del file di input
while (<INPUT>) {
@linea = split;
for ($j = 0; $j < $count; $j++) {
    $lvect[$j] = "NULL";
}
# Se sono valide relazione e soglia
if ( (@linea[1] ne "NORel") && (@linea[3] >= $soglia) &&
    (FreqOk(@linea[0], @linea[4])) ) {

    $lvect[$lffhash{@linea[1]}] = @linea[2];

    # Calcola il numero di volte che l'esempio verrà ripetuto al fine
    # di normalizzarne il peso per C4.5

    $ripetizioni=int(($mcm)*(@linea[3]+0.00001));
    while ($ripetizioni != 0) {
$wrtlines++;
for ($j = 0; $j < $count; $j++) {
    if (@linea[4] eq "Pos") {
print OUTPUT2G $lvect[$j].", ";

```

```
    }
    else {
print OUTPUT2B $lvect[$j].", ";
    }
}
if (@linea[4] eq "Pos") {
    print OUTPUT2G @linea[4]."\n";
    $gtot++;
}
else {
    print OUTPUT2B @linea[4]."\n";
    $btot++;
}
if ($modex ne "-q") {
    msgout int(($wrtlines*100)/($numex*$mcm))."% \r";
}
}
$ripetizioni--;
}
}
}
msgout "FLT-L: Scrittura eseguita.\n";
print "FLT-L: Positivi: $gtot Negativi: $btot\n";
}

# Esecuzione in modalita' ** SINTATTICA **

elsif ($context eq "-s") {

    print "FLT-S: Generazione esempi\n";

    # bffhash: tabella hash per per tenere traccia delle features considerate

    my %bffhash;
    $count = 0;
    $righebuone = 0;
    $numex = 0;
    $oldh = "NULL";
    $oldc = "NULL";
    $coin = 0;
    $gtot = 0;
    $btot = 0;
```

```

# Generazione del file .names secondo il formato richiesto da C4.5

msgout "\nFLT-S: Scrittura del file .names...\n";
print OUTPUT1 "Pos, Neg.\n\n";
while (<INPUT>) {
@linea = split;

# Determina il numero di esempi totali.
# Se ha trovato un nuovo esempio incrementa il counter generale

if ( (@linea[0] ne $oldh) ||
      ( (@linea[0] eq $oldh) && (@linea[4] ne $oldc)) ) {
    $numexTot++;
    $oldh = @linea[0];
    $oldc = @linea[4];
    $coin = 1;
}

if ( (@linea[1] ne "NOReI") && (@linea[3] >= $soglia) &&
      (FreqOk(@linea[0], @linea[4])) ) {

    $righebuone++;

    # L'esempio ha almeno una riga valida. Incrementa counter.
    if ($coin == 1) {
$coin = 0;
$numex++;
    }

    #Determinazione della feature
    $feature= @linea[1];

    # Se si tratta di una nuova feature allora viene inserita
    # nella tabella hash insieme al suo indice progressivo

    if(! exists $bfhash{$feature}) {
$bfhash{$feature} = $count++;
print OUTPUT1 $feature.": continuous.\n";
    }
}
}

```

```
}
msgout "FLT-S: Esempi Validi/Totali: ".$numex."/".$numexTot."\n";
printf (RESULTS "Ex. Val/Tot: %5d/%5d\n",$numex, $numexTot);
msgout "FLT-S: Scrittura eseguita.\n";
msgout "\nFLT-S: Numero di features selezionate: ".$ccount."\n";

# Si posiziona all'inizio del file di input.
seek INPUT,0,0;

# Dichiaro una nuova tabella hash per la memorizzazione degli header
my %bhash;

$ccount = 0;

# Generazione del file .data (rappr. ** SINTATTICA **)

$oldh = "NULL";
$oldc = "NULL";
msgout "\nFLT-S: Scrittura del file .data...\n";
$numrighe = 0;
$coin = 0;

# Prepara un feature vector nullo
for ($j = 0; $j < $ccount; $j++) {
$bvect[$j] = 0;
}

while (<INPUT>) {
@linea = split;

# Considera il precedente esempio letto dal file. Se era stato
# considerato valido allora lo elabora e lo salva sul file in uscita

if ( ($coin == 1) && ($oldh ne "NULL") && (
(@linea[0] ne $oldh) ||
( (@linea[0] eq $oldh) && (@linea[4] ne $oldc)) ) ) {

$coin = 0;
# Calcola la somma delle componenti del vettore esempio
$somma = 0;
for ($j = 0; $j < $ccount; $j++) {
```

```
$somma += $bvect[$j];
}

# Scrive su file il vettore esempio normalizzato
for ($j = 0; $j < $count; $j++) {
if ($j != 0) {
    if ($bvect[$count] eq "Pos") {
print OUTPUT2G ", ";
    }
    else {
print OUTPUT2B ", ";
    }
}
if ($bvect[$count] eq "Pos") {
    print OUTPUT2G ($bvect[$j]/$somma);
}
else {
    print OUTPUT2B ($bvect[$j]/$somma);
}
}

# Scrive la classificazione relativa (Pos/Neg)
if ($bvect[$count] eq "Pos") {
print OUTPUT2G ", $bvect[$count].\n";
$gtot++;
}
else {
print OUTPUT2B ", $bvect[$count].\n";
$btot++;
}

# Pulisce il feature vector per un nuovo esempio
for ($j = 0; $j < $count; $j++) {
$bvect[$j] = 0;
}
}

# Considera la riga corrente. Se appartiene ad un esempio valido
# allora la prende in considerazione ed incrementa il valore
# cumulativo della feature corrispondente
```

```

if ( (@linea[1] ne "NORe1") && (@linea[3] >= $soglia) &&
    (FreqOk(@linea[0], @linea[4])) ) {
    $numrighe++;
    $coin = 1;

    # Stampa percentuali di avanzamento
    if ($modex ne "-q") {
msgout int(($numrighe*100)/($righebuone))."% \r";
    }
    $feature = @linea[1];

    # Calcola il numero di occorrenze della feature
    $bvect[$bftest{$feature}] += @linea[3];
    $bvect[$count] = @linea[4];
}
$oldh = @linea[0];
    $oldc = @linea[4];
}
msgout "FLT-S: Scrittura eseguita.\n";
print "FLT-S: Positivi: $gtot Negativi: $btot\n";
}

# Esecuzione in modalita' ** FREQUENZA **

if ($context eq "-f") {

    print "FLT-F: Generazione esempi\n";

    $numex = 0;
    $plaustot = 0;
    $freq = 0;
    $oldh = "NULL";
    $oldc = "NULL";
    $occtot = 0;
    $gtot = 0;
    $btot = 0;

    # Generazione del file .names secondo il formato richiesto da C4.5

msgout "\nFLT-F: Scrittura del file .names...\n";

```

```

print OUTPUT1 "Pos, Neg.\n\n";
print OUTPUT1 "Freq: continuous\n";
msgout "FLT-F: Scrittura eseguita.\n";

msgout "\nFLT-F: Scrittura del file .data...\n";

$numexTot++;
while (<INPUT>) {
@linea = split;

# Se ha un termine da scrivere, lo fa.
# Cioè ha freq positiva, ed è cambiata la testa xor la classe.

if ( ($oldh ne "NULL") && (@linea[0] ne $oldh) ||
      (@linea[0] eq $oldh) && (@linea[4] ne $oldc) ) {

    $numexTot++;
    if ($freq > 0) {
$numex++;
$fpvect[$numex] = $freq;
$cpvect[$numex] = $oldc;
$freq = 0;
    }
}

# Considera ora l'esempio corrente

$plaustot += @linea[3];
if ($plaustot >= 1-$epsilon) {
    $plaustot = 0;
    if ( (@linea[1] ne "NORe1") && (@linea[3] >= $soglia) &&
          (FreqOk(@linea[0], @linea[4])) ) {
$occtot++;
$freq++;
    }
}
$oldh = @linea[0];
$oldc = @linea[4];
}
for ($i = 1; $i <= $numex; $i++) {
if ($cpvect[$i] eq "Pos") {

```

```

        print OUTPUT2G $fpvect[$i]*1000/$occtot.", ".$cpvect[$i]."\n";
        $gtot++;
    }
else {
    print OUTPUT2B $fpvect[$i]*1000/$occtot.", ".$cpvect[$i]."\n";
    $btot++;
}
}
msgout "FLT-F: Scrittura eseguita.\n";
msgout "FLT-F: Esempi Validi/Totali: ".$numex."/".$numexTot."\n";
msgout "FLT-F: Occorrenze totali: ".$occtot."\n";
printf (RESULTS "Ex. Val/Tot: %5d/%5d\n",$numex,$numexTot);
print "FLT-F: Positivi: $gtot Negativi: $btot\n";
}

# Analisi sulla frequenza degli header nel file di ingresso

elsif ($context eq "-a") {

    if (! -r $gfile) { die "Can't read input $gfile\n";}
    if (! -f $gfile) { die "Input $gfile is not a plain file\n";}
    open (AINPUT, "<$gfile") || die "Can't input $gfile $!";

    my %fmhash;
    my @fvect;
    my $card = 0;
    my $extot = 0;
    my $exflag = 0;
    my $oldhead = "NULL";
    my $word = "NULL";

    # Scandisce il file in ingresso raccogliendo le info sulla frequenza
    while (<AINPUT>) {
@linea = split;

# Si adatta a lavorare eventualmente con esempi positivi e negativi
# fusi assieme.

if ( (@linea[4] eq "Pos") || (@linea[4] eq "Neg") ) {
    $word = @linea[0]."-".@linea[4];
}

```

```

else {
    $word = @linea[0];
}

$exflag += @linea[3];
if ($exflag >= 1-$epsilon) {
    $exflag = 0;
    if ( (@linea[1] ne "NORel") && (@linea[3] >= $soglia) ) {
$extot++;
if (! exists $fmhash{$word}) {
    $card++;
    $fmhash{$word} = $card;
}
$fvect[$fmhash{$word}]++;
}
}

# Determina le frequenze massima e minima riscontrate
$freqmax = $fvect[1];
$freqmin = $fvect[1];
for ($i = 1; $i <= $card; $i++) {
if ($fvect[$i] > $freqmax) {
    $freqmax = $fvect[$i];
}
if ($fvect[$i] < $freqmin) {
    $freqmin = $fvect[$i];
}
}

seek AINPUT,0,0;
open (AOUTPUT, ">./out.freq") || die "Can't output $outf.names $!";

# Genera un file contenente tutte le teste con le rispettive frequenze

print AOUTPUT "Frequenze assolute e per mille con MaxAmbig=$maxambig\n\n";
while(<AINPUT) {
@linea = split;

if ( (@linea[4] eq "Pos") || (@linea[4] eq "Neg") ) {
    $word = @linea[0]."-".@linea[4];
}
}

```

```
else {
    $word = @linea[0];
}

$floc = $fvect[$fmhash{$word}];
if ($floc > 0) {
    printf (AOUTPUT "%20s %4d\t%.3f\n", $word, $floc,
        $floc*1000/$extot);
    $fvect[$fmhash{$word}] = 0;
}

}

# Stampa a video informazioni riepilogative
printf ("\n\tVincolo ambig: %5d\n", $maxambig);
print "\n\tANALISI DEL FILE:\n";
printf ("\tOcc. totali : %5d\n", $extot);
printf ("\tOcc. distinte: %5d\n", $card);
printf ("\tFreq. minima : %5d=%7.3f%%%\n", $freqmin, $freqmin*1000/$extot);
printf ("\tFreq. massima: %5d=%7.3f%%%\n", $freqmax, $freqmax*1000/$extot);
printf ("\tFreq. media : %5.1f=%7.3f%%%\n", $extot/$card, 1000/$card);
print "\n\n\tAnalisi dettagliata scritta sul file out.freq\n\n";

close AINPUT;
close AOUTPUT;
}

# Chiusura dei file aperti.

if ($context ne "-a") {
    close INPUT;
    close OUTPUT1;
    close OUTPUT2G;
    close OUTPUT2B;
}

exit;
```

A.2 Codice di *iter.pl*

```
#!/usr/bin/perl

#
# Invoca iterativamente n esecuzioni di c4.5, facendo variare "per slot"
# il training-set ed il test-set.
#
# Il mescolamento iniziale del file contenente gli esempi è ottenuto in
# tre passi:
#
# 1 - viene inserito un numero casuale all'inizio di ciascuna riga
# 2 - viene invocato il sort di sistema sul file così predisposto
# 3 - viene eliminato il numero casuale all'inizio di ciascuna riga
#

sub modulo ($$) {
    my ($x,$y) = @_;
    if ($x == $y) {
return $x;
    }
    return $x-int($x/$y)*$y;
}

sub Randomize($) {

    #Imposta seed iniziale per generatore pseudocasuale
    srand (time);

    my ($infile) = @_;

    if (! -r $infile.".data") {die "Can't read input $infile.data\n";}
    if (! -f $infile.".data") {die "Input $infile.data is not a plain file\n";}
    open (INPUT, "<$infile.data") || die "Can't input $infile $!";
    open (OUTPUT, ">$infile.num") || die "Can't open $outfile $!";

    # Passo 1: Le righe vengono numerate con identificativi generati casualm.
```

```
    while (<INPUT>) {
print OUTPUT int((rand(2**32-1)))."\t".$_;
    }

    close INPUT;
    close OUTPUT;

    # Passo 2: viene invocato il sort di sistema sul file ottenuto al
    # passo precedente. Le righe risultano dunque mescolate casualmente.
    $code=system "sort -n -o $infile.sorted $infile.num";
    if ($code != 0) {
print"ITER : Errore nel sort! Exit-Code: ".$code."\n";
    }

    #Rimozione del file temporaneo utilizzato
    system "rm -f $infile.num";

    open(SORTED, "<$infile.sorted");
    open(FINAL, ">$infile.final");

    $count=0;

    # Passo 3: vengono eliminati i numeri casuali inseriti al passo 1.
    while (<SORTED>) {
$count++;
@linea=split;
#Stampa su un altro file il contenuto del file ordinato senza includere
#il numero casuale.
print FINAL join(" ",@linea[1..$#linea]);
print FINAL "\n";
    }

    close SORTED;
    close FINAL;

    #Rimozione del file temporaneo utilizzato
    system "rm -f $infile.sorted";

}

```

```
( $file, $nblocchi, $mode) = @ARGV;

print"ITER : Mescolamento in corso\n";

Randomize($file."g");
$countg = $count;

Randomize($file."b");
$countb = $count;

# Calcolo del numero di righe in ciascuno slot
$numlinesg=int($countg/$nblocchi);
$numlinesb=int($countb/$nblocchi);

if (! -r $file."g.final") { die "Can't read input $file"."g.final\n"; }
if (! -f $file."g.final") { die "Input $file"."g.final is not a plain file\n";}
open (INFILEG, "<$file"."g.final");

if (! -r $file."b.final") { die "Can't read input $file"."b.final\n"; }
if (! -f $file."b.final") { die "Input $file"."g.final is not a plain file\n";}
open (INFILEB, "<$file"."b.final");

open (RESULTS, ">>results");

if ($mode eq "-j") {
    $start = $nblocchi-1;
}
else {
    $start = 1;
}

for ($i = 1; $i <= $nblocchi; $i++) {
    for ($j = $start; $j < $nblocchi; $j++) {

        $writteng = 0;
        $datalinesg = 0;
        $testlinesg = 0;

        $writtenb = 0;
        $datalinesb = 0;
        $testlinesb = 0;
```

```

seek INFILEG,0,0;
seek INFILEB,0,0;

open (OUTDATA, ">$file-$i-$j.data");
open (OUTTEST, ">$file-$i-$j.test");

while (<INFILEG>) {
    #Se l'elemento che stiamo per scrivere deve essere considerato
    #come esempio per il test
    if ($writteng >= ($i-1)*$numlinesg && $writteng < $i*$numlinesg) {
print OUTTEST $_;
    $testlinesg++;
    }
    #Se l'elemento deve essere usato per l'apprendimento
    else {
print OUTDATA $_;
    $datalinesg++;
    }
    $writteng++;
}
while (<INFILEB>) {
    if ($writtenb >= ($i-1)*$numlinesb && $writtenb < $i*$numlinesb) {
print OUTTEST $_;
    $testlinesb++;
    }
    elsif (($i+$j) <= $nblocchi
    && $writtenb < modulo(($i+$j),$nblocchi)*$numlinesb &&
    $writtenb >= modulo($i,$nblocchi)*$numlinesb) {
print OUTDATA $_;
    $datalinesb++;
    }
    elsif (($i+$j) > $nblocchi
    && ($writtenb < modulo(($i+$j),$nblocchi)*$numlinesb ||
    $writtenb >= modulo($i,$nblocchi)*$numlinesb)) {
print OUTDATA $_;
    $datalinesb++;
    }
    $writtenb++;
}
printf ("ITER : (%d,%d) - TRAIN.: p:%5d n:%5d t:%5d", $i, $j, $datalinesg, $datalinesb);

```

```
printf (" - TEST: p:%5d n:%5d t:%5d\n", $testlinesg, $testlinesb, $testlinesg+$testlinesb)

close OUTDATA;
close OUTTEST;

# Rinominazione del file .names da fornire a c4.5
system "cp $file.names $file-$i-$j.names";

# Esecuzione di c4.5, i-j-esima iterazione
system "../R8/Src/c4.5 -f $file-$i-$j -u -n >> out.c4.5-trees";

# Rimozione dei file usati da c4.5
system "rm -f $file-$i-$j.data";
system "rm -f $file-$i-$j.test";
system "rm -f $file-$i-$j.names";
system "rm -f $file-$i-$j.unpruned";
}

if (! -r "out.c4.5-errors") { die "Can't read input file out.c4.5.errors\n";}
if (! -f "out.c4.5-errors") { die "Input file out.c4.5-errors is not a plain file\n";}

# Elab. risultati ottenuti nelle varie iterazioni e aggiornamento
# del report finale

open (STATS, "<out.c4.5-errors");

for ($j = $start; $j < $nblocchi; $j++) {
$_ = <STATS>;
@lvalues = split;
$dim[$j] += $lvalues[0];
$err[$j] += $lvalues[2];
}

#Rimozione file temporaneo - DEVE rimuoverlo pena il fallimento di BATCH
system "cat out.c4.5-errors >> out.c4.5-allerrors";
system "echo >> out.c4.5-allerrors";
system "rm -f out.c4.5-errors";

close STATS;

}
```

```
close INFILEG;
close INFILEB;

for ($j = $start; $j < $nblocchi; $j++) {
    $dim[$j] = $dim[$j]/$nblocchi;
    $err[$j] = $err[$j]/$nblocchi;
    #print "DIM-$j: $dim[$j] - ERR-$j: $err[$j]\n";
    print RESULTS "DIM-$j: $dim[$j] - ERR-$j: $err[$j]\n";
}

print "\n";
print RESULTS "\n";

# Rimozione file generale mescolato contenente tutti gli esempi
system "rm -f $file"."g.final";
system "rm -f $file"."b.final";

close RESULTS;

exit;
```

A.3 Esempio di script *batch*

```
#!/bin/sh

# Linea di comando: batch MaxAmbig Slots Fmin Fmax

rm -f out.c4.5-allerrors
rm -f out.c4.5-trees

echo
date
echo "Max ambiguità      : $1" > results
echo "Numero di slot    : $2" >> results
echo "Frequenza minima  : $3" >> results
```

```
echo -e "Frequenza massima: $4\n" >> results
echo

#echo "BATCH: Esperimento 1/8: local/Lifuv"
#echo -n "SET=Lifuv - RAPPR=L - " >> results
#./filtro.pl -l ../data/Lifuv/good.txt ../data/Lifuv/bad.txt $1 $3 $4 local -q
#./iter.pl local $2

echo "BATCH: Esperimento 2/8: global/Lifuv"
echo -n "SET=Lifuv - RAPPR=G - " >> results
./filtro.pl -g ../data/Lifuv/good.txt ../data/Lifuv/bad.txt $1 $3 $4 global -q
./iter.pl global $2

echo "BATCH: Esperimento 3/8: syntax/Lifuv"
echo -n "SET=Lifuv - RAPPR=S - " >> results
./filtro.pl -s ../data/Lifuv/good.txt ../data/Lifuv/bad.txt $1 $3 $4 syntax -q
./iter.pl syntax $2

echo "BATCH: Esperimento 4/8: frequency/Lifuv"
echo -n "SET=Lifuv - RAPPR=F - " >> results
./filtro.pl -f ../data/Lifuv/good.txt ../data/Lifuv/bad.txt $1 $3 $4 frequency -q
./iter.pl frequency $2

#echo "BATCH: Esperimento 5/8: local/Zerolex"
#echo -n "SET=Zerolex - RAPPR=L - " >> results
#./filtro.pl -l ../data/Zerolex/good.txt ../data/Zerolex/bad.txt $1 $3 $4 local -q
#./iter.pl local $2

echo "BATCH: Esperimento 6/8: global/Zerolex"
echo -n "SET=Zerolex - RAPPR=G - " >> results
./filtro.pl -g ../data/Zerolex/good.txt ../data/Zerolex/bad.txt $1 $3 $4 global -q
./iter.pl global $2

echo "BATCH: Esperimento 7/8: syntax/Zerolex"
echo -n "SET=Zerolex - RAPPR=S - " >> results
./filtro.pl -s ../data/Zerolex/good.txt ../data/Zerolex/bad.txt $1 $3 $4 syntax -q
./iter.pl syntax $2

echo "BATCH: Esperimento 8/8: frequency/Zerolex"
echo -n "SET=Zerolex - RAPPR=F - " >> results
./filtro.pl -f ../data/Zerolex/good.txt ../data/Zerolex/bad.txt $1 $3 $4 frequency
```

```
./iter.pl frequenze $2  
  
echo  
date  
echo -e "\nREPORT"  
echo -e "-----\n"  
cat results
```

Bibliografia

- [1] R. Basili, M.T. Paziienza, F.M. Zanzotto (2001) – Modelling syntactic context in automatic term extraction .
- [2] R. Basili, M.T. Paziienza, F.M. Zanzotto (2000) – Customizable Modular Lexicalized Parsing – *Proceedings of IWPT 2000*– Trento (Italy), pagg. 41–52 .
- [3] Ralf D. Brown (1998) – Automatically-Extracted Thesauri for Cross-Language IR: When Better is Worse –*Proceedings of Computerm'98*– pagg. 15–21 .
- [4] A. Condamines e J. Rebeyrolle (1998) – CTKB: A Corpus-based approach to a Terminological Knowledge Base –*Proceedings of Computerm'98*– pagg. 29-35 .
- [5] L. Davidson, J. Kavanagh, K. Mackintosh, I. Meyer, D. Skuce (1998) – Semi-automatic Extraction of Knowledge-rich Context from Corpora –*Proceedings of Computerm'98*– pagg. 50–56 .
- [6] Dekang Lin –Extracting Collocations from Text Corpora –*Proceedings of Computerm'98*– pagg. 57–63 .

- [7] K. Kageura, M. Yoshioka, T. Koyama, T. Nozue – Towards a Common Testbed for Corpus-Based Computational Terminology – *Proceedings of Computerm'98*– pagg. 81–85.
- [8] M. P. Oakes e C. D. Paice –Term Extraction for Automatic Abstracting – *Proceedings of Computerm'98*– pagg. 91–95.
- [9] S.Russell e P. Norvig (1998)– Intelligenza artificiale, un approccio moderno– UTET Libreria
- [10] Christian Jacquemin (2001)– Spotting and discovering terms through natural language processing – The MIT Press
- [11] Pat Langley (1996)– Elements of machine learning– Morgan Kaufmann Publishers, Inc.
- [12] J. Hopcroft e J. Ullman (1979)– Introduction to Automata Theory, Languages, and Computation – Addison Wesley Publishing Company
- [13] T. Allen (1998) – Natural Language Understanding – Benjamin Publisher.
- [14] J. R. Quinlan (1993) – C4.5: Programs for Machine Learning – Morgan Kaufman.